(2)

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

CALNPS
COMPUTER ANALYSIS LANGUAGE NAVAL POSTGRADUATE SCHOOL VERSION

by

Leonard L. Langford, Jr.

June 1989

Thesis Advisor Gilles Cantin

Approved for public release; distribution is unlimited.

Unclassified

security classification of this page

REPORT DOCUMENTATION PAGE

1a Report Security Classification Unclassified			1b Restrictive Markings		
2a Security Classification Authority			3 Distribution Availability of Report Approved for public release; distribution is unlimited.		
2b Declassification Downgrading Schedule					
4 Performing Organization Report Number(s)			5 Monitoring Organization Report Number(s)		
6a Name of Performing Organization Naval Postgraduate School	6b Office Symbol (if applicable) 34		7a Name of Monitoring Organization Naval Postgraduate School		
6c Address (city, state, and ZIP code) Monterey, CA 93943-5000			7b Address (city, state, and ZIP code) Monterey, CA 93943-5000		
8a Name of Funding Sponsoring Organization	8b Office Symbol (if applicable)		9 Procurement Instrument Identification Number		
8c Address (city, state, and ZIP code)			10 Source of Funding Numbers Program Element No	Project No	Task No / Work Unit Accession No
11 Title (include security classification) CALNPS COMPUTER ANALYSIS LANGUAGE NAVAL POSTGRADUATE SCHOOL VERSION					
12 Personal Author(s) Leonard L. Langford, Jr.					
13a Type of Report Master's Thesis	13b Time Covered From To		14 Date of Report (year, month, day) June 1989	15 Page Count 114	
16 Supplementary Notation The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.					
17 Cosati Codes			18 Subject Terms (continue on reverse if necessary and identify by block number) word processing, Script, GML, text processing.		
Field	Group	Subgroup			

19 Abstract (continue on reverse if necessary and identify by block number)

The Computer Analysis Language (CAL) Program was originally written by Professor Edward L. Wilson of the University of California at Berkeley as a teaching tool for structural analysis. The program was modified for use on the Naval Postgraduate School (NPS) mainframe (IBM 360 67) in 1979 by Lawrence B. Elliott, Lieutenant Commander, U.S.N. The modified version was called CALNPS. In 1982, Warren L. Roberts, Lieutenant, U.S.N., integrated the Finite Element Analysis Program (FEAP) with CALNPS. This provided a means for the solution of linear and nonlinear, two and three dimensional, and, steady state and transient heat conduction problems. Roberts also generated an interactive "HELP" facility and the code for terminal graphics displays of heat transfer and structural analysis meshes. Since then, changes to the NPS computer system and transition of CALNPS to the VAX computer system have rendered CALNPS unusable in many ways. The "HELP" facility is obsolete. The purpose of this thesis was to bring CALNPS back up to date, rewrite the "HELP" facility, and make the program "user friendly". Also several modifications were added to CALNPS. The graphics capabilities were expanded to include hardcopy options using the Plot10 and Disspla graphics libraries. Two display size options are now available and the user now has the capability to plot curves from data files from within the CALNPS domain.

As CALNPS is a very large program, several of the functions available had not been tested completely and as a result did not work at all or did not work in the manner described in the user's manual. This thesis work included the testing of every command and verifying that they work in accordance with the user's manual. Several problems were discovered and corrected by either changing the FORTRAN code or the instructions or both. The work was focused around use of the VAX computer system.

20 Distribution Availability of Abstract [✓ unclassified unlimited] ☐ same as report ☐ DTIC users	21 Abstract Security Classification Unclassified
22a Name of Responsible Individual Gilles Cantin	22b Telephone (include Area code) (408) 646-2364 / 22c Office Symbol 69Ci

DD FORM 1473,84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete — security classification of this page

Unclassified

CALNPS
Computer Analysis Language
Naval Postgraduate School Version

by

Leonard L. Langford, Jr.
Lieutenant, United States Navy
B.S., Auburn University, 1980

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 1989

Author: [illegible]
Leonard L. Langford, Jr.

Approved by: Gilles Cantin
Gilles Cantin, Thesis Advisor

[illegible]
Anthony Healey, Chairman,
Department of Mechanical Engineering

[illegible]
Gordon E. Schacher,
Dean of Science and Engineering

ABSTRACT

The Computer Analysis Language (CAL) Program was originally written by Professor Edward L. Wilson of the University of California at Berkeley as a teaching tool for structural analysis. The program was modified for use on the Naval Postgraduate School (NPS) mainframe (IBM 360/67) in 1979 by Lawrence B. Elliott, Lieutenant Commander, U.S.N. The modified version was called CALNPS. In 1982, Warren L. Roberts, Lieutenant, U.S.N., integrated the Finite Element Analysis Program (FEAP) with CALNPS. This provided a means for the solution of linear and nonlinear, two and three dimensional, and, steady state and transient heat conduction problems. Roberts also generated an interactive "HELP" facility and the code for terminal graphics displays of heat transfer and structural analysis meshes. Since then, changes to the NPS computer system and transition of CALNPS to the VAX computer system have rendered CALNPS unusable in many ways. The "HELP" facility is obsolete. The purpose of this thesis was to bring CALNPS back up to date, rewrite the "HELP" facility, and make the program "user friendly". Also several modifications were added to CALNPS. The graphics capabilities were expanded to include hardcopy options using the Plot10 and Disspla graphics libraries. Two display size options are now available and the user now has the capability to plot curves from data files from within the CALNPS domain.

As CALNPS is a very large program, several of the functions available had not been tested completely and as a result did not work at all or did not work in the manner described in the user's manual. This thesis work included the testing of every command and verifying that they work in accordance with the user's manual. Several problems were discovered and corrected by either changing the FORTRAN code or the instructions or both. The work was focused around use of the VAX computer system.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

A. GENERAL DESCRIPTION

CALNPS is an interpretive language which is designed to manipulate arrays and matrices for the analysis of structures and heat conduction problems. With the integration of the Finite Element Analysis Program (FEAP), CALNPS has the ability to solve linear and nonlinear, steady state and transient, two and three dimensional heat conduction problems involving temperature dependent thermophysical properties and complicated radiation/convection boundary conditions. The original program was designed as an instructional tool for structural analysis and published as [Ref. 1].

Modifications have been added to expand the graphics capabilities to include hardcopy options and two size options. Three commands were added to CALNPS in this thesis work. The ability to solve a cubic equation with real or imaginary coefficients or roots was added and is enabled with the command, CUBIC. The ability to plot curves from existing data files was added and is enabled with the command, XYPLOT. This function utilizes the Plot10 graphics library for the terminal graphics and the Disspla graphics library for the hardcopy options. An online demonstration routine was also added. This required the addition of the new command, DEMO. DEMO provides a menu of example data files that the user can choose to view. The files are complete with graphics displays.

CALNPS can be operated in either the interactive or batch mode. In the batch mode, the command, READ, is used to obtain the existing data file from the user's local directory. In the interactive mode, the user has the capability to save the session using the command, SAVE, at any point. The SAVE command creates a data file with the name supplied by the user (prompted) which contains all of the arrays in storage at the time of issuance. The session can later be recalled with the command, RESUME. The use of the SAVE command is highly recommended when using the interactive mode as input errors to CALNPS frequently result in stoppage of the program and loss of any data not previously saved.

CALNPS can be used to solve simple problems for instructional purposes or more complex problems for research purposes. This thesis work provides an expansion of the sample problem files that previously existed including input and output data files. Some problems are simple to demonstrate the use of the commands and some are more complex to demonstrate the capabilities of CALNPS.

CALNPS was recently implemented on the VAX computer system at NPS. This thesis is the completion of this implementation and the generation of a revised user's manual.

B. OBJECTIVES

The objectives of this thesis are:

(1) Test every CALNPS command in the current manuals with example problems and determine if they work properly.

(2) Make corrections to CALNPS code and or user's manual as necessary. During the course of the work several problems were discovered and corrected.

(3) Expand the graphics capabilities. The graphics capabilities were limited to the use of the print screen option (on the VAX computer) at the beginning of this work.

(4) Rewrite the "HELP"facility. The online "HELP" facility was obselete and did not accurately apply to the VAX computer.

(5) Revise the user's manual. The same situation existed with the user's manual as with the "HELP" facility. The user's manual still contained references to FORTRAN card decks.

II. ORGANIZATION OF CALNPS

This chapter provides a description of the internal organization of CALNPS. CALNPS consists of a main program and five operational groups of subroutines. CALNPS is also linked to the PLOT10 and DISSPLA graphics libraries. The graphics libraries will not be discussed in detail in this work. The average user does not need to be concerned with the workings of these libraries as there is no required user interface with them. The user who is interested in (and has the privilege to do so) modifying CALNPS should refer to the appropriate user's manual as necessary.

A. MAIN PROGRAM

The main program as referred to here consists of the actual main program and the subroutines that accomplish data management and interpretation of the input operation commands. The subroutine CAL1 recognizes the input operation command and determines which group the command and its related subroutines reside in. Before the program branches to the appropriate group, CAL1 performs a series of data management functions.

The subroutine INPUT checks to see if the input operation command is part of a loop or a new command. If it is a new command, then the subroutines OPREAD and RCARD are called to interpret the input data associated with the operation command. For example, matrix names, dimensions, numerical data, etc. If the command is part of a loop then the OPREAD RCARD operations continue until all of the loop operations have been read and stored. The subroutines INPUT and OPSTOR work in conjunction to carry out the loop operations properly.[Ref. 2. pp. 34-38]

Dynamic dimensioning is used to store all arrays and matrices in CALNPS. For a detailed discussion of dynamic dimensioning refer to chapter II. of Elliott's thesis[Ref. 2: pp. 11-18]. The subroutine LIST reserves the storage of new matrices and prepares the directories. The subroutine FIND is called later to locate the desired array and its directory when required by the current operation command.

When no longer required, arrays and matrices are deleted by the subroutine DELETE. DELETE is also called prior to creating a new matrix or array to prevent duplication of names. If a command creates an array or matrix with the same name as an existing one, then the existing one is automatically deleted. The user is informed of the deletion after the fact and there is no way to recover the deleted array or matrix unless it was saved with the command, SAVE. The user must keep track of names assigned to arrays and matrices.

This section provides a brief discussion of the data management process within CALNPS. Figure 1 shows the data flowpath between the subroutines of the main program. Only the major functions have been discussed here. For a more detailed discussion of this process, refer to chapter III. of Elliot's thesis [Ref. 2: pp. 34-43].

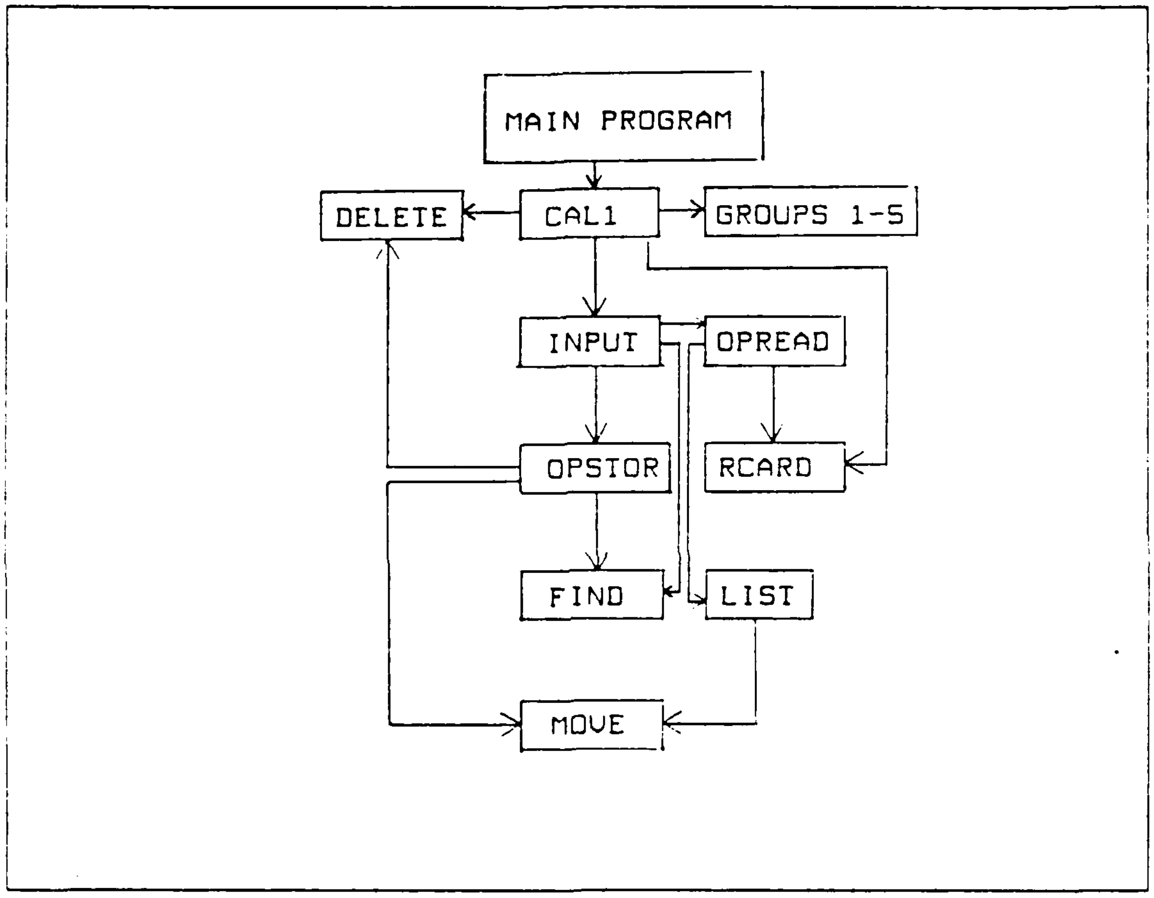


Figure 1. Data Flowpath Within Main Program Subroutines

B. GROUP ONE

Group One consists of the subroutines necessary to support general commands, general matrix commands, user supplied operations, and looping operations. Each of these three functions will be explained and the function of each command will be listed in the following discussion.

1. General Commands

General commands perform standard functions that allow the user to interface and control CALNPS. The General Commands and their functions are listed below in alphabetical order:

CALNPS	Provides a list of CALNPS commands when issued from the "HELP" facility.
DEMO	Runs a sample problem as chosen from a menu by the user.
HELP	Enables the interactive HELP facility.
LABEL	Reads and prints comment lines.
LIST	Lists arrays currently in storage and the amount of storage used.
NO	Suppresses all printing of output.
READ	Directs reading of input from terminal or data file.
RESUME	Reads a saved file into memory.
SAVE	Saves all arrays currently in storage to a data file.
START	Initializes CALNPS for a new problem by deleting all arrays in storage.
STOP	Terminates CALNPS operations.
WRITE	Directs output to a file or to the screen.
YES	Restores printing of output. (Negates the NO operation.)

2. General Matrix Commands

The General Matrix Commands perform standard matrix manipulations. The specific commands and their functions are listed in alphabetical order as follows:

ADD	Adds two matrices.
ADDSM	Adds a smaller matrix to a designated position in a larger matrix.
COSEL	Evaluates the cosine of each element of a matrix.
CUBIC	Evaluates the roots of a cubic equation.
DELETE	Deletes a matrix from storage.
DUP	Duplicates a designated matrix.
DUPDG	Creates a row matrix from diagonal of an existing matrix.
DUPSM	Creates a smaller matrix by extracting designated positions from a larger matrix.
INVEL	Inverts each element of a matrix.
LOAD	Loads a real matrix.
LOADI	Loads an integer array.

LOG	Evaluates the natural log of each element of a matrix.
MAX	Evaluates the maximum value of each row of a matrix.
MULT	Multiplies two matrices.
NORM	Evaluates matrix norms.
PRINT	Prints a matrix.
PROD	Evaluates the product of all the terms of a matrix.
SCALE	Multiplies a matrix by a scaler.
SINEL	Evaluates the sine of each element of a matrix.
SOLVE	Solves a system of linear equations.
SQREL	Evaluates the square root of each element of a matrix.
STODG	Stores a row of a matrix on the diagonal of another.
STOSM	Stores a submatrix within a larger matrix.
SUB	Subtracts two matrices.
SUBSM	Subtracts a small matrix from a designated area in a larger matrix.
TRAN	Generates the transpose of a matrix.
ZERO	Creates a null or a unit matrix.

3. User Supplied Operations

There are currently two areas set aside in CALNPS for user supplied subroutines. The commands to execute a subroutine placed in one of these areas are as follows:

USERA

USERB

4. Looping Operations

CALNPS has a five level looping ability [Ref. 3: pp. 75]. The specific looping operations commands and their functions are listed in alphabetical order below:

LOOP	Initiates the looping operation.
NEXT	Signifies the end of a loop.
SKIP	Causes a specified number of operations to be skipped.

C. GROUP TWO

The Group Two subroutines support the evaluation of structural analysis problems. This group independently provides a static solution, however, when used in conjunction with Group Three, a dynamic solution is provided. The specific Group Two Commands and their functions are listed in alphabetical order below:

1. Static Analysis Commands

ADDK	Adds element stiffness matrix to total stiffness matrix.
ADDSF	Forms total stiffness and mass matrices.
BEAM	Forms the element stiffness, mass, and force-displacement transformation matrices for a three dimensional beam member.
BOUND	Specifies displacement boundary conditions.
DISPL	Prints joint displacements.
FORCE	Calculates and prints member forces.
FRAME	Forms the stiffness matrix for a two dimensional frame member.
LOADS	Forms the load matrix.
MEMFRC	Calculates member forces in a two dimensional system.
NODES	Creates an array containing the coordinates for all joints in a structural system.
PLANE	Calculates the element stiffness, mass, and stress-displacement matrices for three to eight node isoparametric elements.
SLOPE	Forms the stiffness matrix for a beam or column from the slope deflection equations.
TRUSS	Forms the element stiffness, mass, and force-displacement matrices for three dimensional truss members.

D. GROUP THREE

The Group Three subroutines work in conjunction with Group Two to solve dynamic structural analysis problems. The specific Group Three commands and their functions are listed in alphabetical order below:

1. Dynamic Analysis Commands

DYNAM	Evaluates uncoupled equations of motion by mode superposition method.
EIGEN	Evaluates mode shapes and frequencies.
FUNG	Generates values of a specified function at equal time intervals.
PLOT	Generates a printer plot of selective rows of a designated matrix.
STEP	Evaluates the dynamic response of a structural system by integrating the dynamic equilibrium equations.

E. GROUP FOUR

The Group Four subroutines support the solution of heat transfer problems. The specific Group Four commands and their functions are listed in alphabetical order below:

1. **Heat Transfer Commands**

ADTIM	Advances the time in a heat transfer problem.
CALC	Solves time independent problems for temperature
CCAP	Forms a consistent capacitance matrix.
CONV	Performs a temperature convergence test.
COORD	Creates an array containing the coordinates of all nodes in a heat transfer system.
CTEMP	Inputs constant temperature boundary restraint data.
DTIM	Set time step increment.
EIGV	Evaluates the dominant eigenvalue and eigenvector of the current heat transfer conductance matrix.
ELCON	Creates an array containing the element connectivity.
FORM	Forms the flux vector.
HTXFR	Initializes the heat transfer problem.
LCAP	Forms a lumped capacitance matrix.
ODE	Solves first order ordinary differential equations for heat transfer problems.
PRLD	Provides for proportional loading of heat transfer systems.
PROF	Establishes the profile of the equations for solution of the problem.
PROMPT	Suppresses or restores user input prompts.
PROP	Inputs material property data.
PTEMP	Prints nodal temperatures.
SYMC	Forms the symmetric conductance matrix.
TOL	Sets the solution convergence tolerance.
USYMC	Forms the unsymmetric conductance matrix.

F. GROUP FIVE

The Group Five subroutines support the graphics capabilities of CALNPS. The specific Group Five commands and their functions are listed below alphabetically:

1. Graphics Commands

GRAPH	Initializes the graphics package.
PLHX	Plots two and three dimensional heat transfer analysis meshes.
PLST	Plots two and three dimensional structural analysis meshes.
TITLE	Allows the user to input a title of up to three lines in length.
XYPLOT	Plots up to two curves on a plot of data read from a data file.

III. MODIFICATIONS TO CALNPS

This chapter describes the modifications that were made to the existing version of CALNPS during this thesis work. The modifications range in complexity from a simple change in a format statement to a complex change in the FORTRAN code. Modifications were necessary to correct functions that did not work properly and to expand the capabilities of CALNPS. In total, 17 changes were made to the CALNPS code. Within the code these changes are clearly marked with the author's initials and by comment lines.

A. CHANGE 1: SAVE AND RESUME COMMANDS

The SAVE and RESUME commands did not work properly. The option to save a heat transfer file did not work at all. The problem was found to be that two subroutines existed which could be accessed by CALNPS depending on the value of N1 entered by the user. The default was zero which directed the program to the SAVE subroutine. This worked fine as long as the problem at hand was a structural analysis problem. The heat transfer option requires N1 to be equal to two which directs the program to the SAVE2 subroutine. The SAVE2 subroutine contained an error in the file opening statement. The file status was set as "NEW", however, this statement is used by both the SAVE and RESUME commands. This prevented the RESUME command from reading an existing heat transfer data file. The status was changed to "UNKNOWN", which is compatible with both the SAVE and RESUME commands. The original SAVE subroutine was actually obsolete due to the changes made to this command in Robert's thesis [Ref. 3: pp. 20]. The SAVE subroutine was commented out completely and all input correctly rerouted to the SAVE2 subroutine. The proper use of the SAVE and RESUME commands is as follows:

SAVE,N1 and RESUME,N1

where:

N1 = 1	A structural problem is being saved or resumed.
N1 = 2	A heat transfer problem is being saved or resumed.

The default was changed to N1 = 1.

B. CHANGE 2: SOLVE COMMAND

This change is for clarity purposes only, no computational code changes were made. The command SOLVE,M1,M2,N1,N2 solves the matrix equation $AX = B$ where M1 is the name of the A matrix and M2 is the name of the B matrix. The information provided to the user concerning what was happening to the matrices involved was unclear. The program and the "HELP" file both state that matrix M1 is triangularized. This is true, however, it is also stored as a compact lower upper (L U) form, where L is a lower unit matrix which is obtained from a forward reduction and a backward substitution procedure. The signs of the off diagonal terms are reversed for convenience only. Write statements were added to provide this information to the user.

When $N1 = 2$ the system is transformed as stated above, however, the program indicated that the system was not solved, which contradicted the information in the "HELP" file. The system is in fact solved and the solution is stored as matrix M2. A write statement was added to provide this information correctly to the user.

C. CHANGE 3: LOG COMMAND

The LOG command was discovered to be completely inoperable. The response to the LOG command was "operation undefined or blank", which meant that CALNPS did not recognize this command as a valid command. It was discovered that the command, LOG,M1, as listed in the "HELP" file was not the same as that in the code. The code required the command, DLOG,M1, to carry out the required operation, which is to replace each term in the matrix M1 with the natural log of each term. A test of the command, DLOG,M1, was conducted to verify that it worked properly. The decision was made to change the code to recognize the command as stated in the "HELP" file rather than change the "HELP" file to match the code. This was done to make the command more easily recognizable to the user.

D. CHANGE 4: COSEL AND SINEL COMMANDS

The ability to calculate the sine and cosine of each term of a matrix was discovered in the CALNPS code. The commands to utilize these functions were not listed in the "HELP" file or in the user's manual. These functions are part of the Group 1 subroutines. These commands were verified to work properly and the "HELP" file and user's manual were updated to reflect these commands and their use.

E. CHANGE 5: PROD COMMAND

The PROD command calculates the product of the elements of a matrix. The routine was found to be calculating a product that was a factor of ten high. The value of the exponent in the code was discovered to be initialized as one instead of zero. This change solved the problem.

The output of the command did not always produce correct scientific notation. For example, the output would print as 10.00E+00 rather than 1.00E+01. A change was implemented to insure consistent and correct output format.

F. CHANGE 6: GRAPH COMMAND

The GRAPH command initializes the graphics subroutines of CALNPS. After issuing the command, GRAPH, the user was required to respond to two questions before CALNPS would proceed to the graphics operations. This is necessary when the user is working from an IBM mainframe terminal, however, on the VAX computer system the responses are always the same. The first question is: "Are you at one of the following graphics terminals (yes or no)? PLOT10 compatible terminal or IBM 3277 dual screen." The answer is always "yes" from a VAX terminal. The second question is "Enter terminal code: 1=PLOT10 compatible or 2=IBM dual screen." The response is always one from a VAX terminal. The subroutine GRAFI was modified to delete these response requirements and to allow CALNPS to proceed to the graphics operations after receiving the command, GRAPH. This change does not apply to the IBM mainframe version of CALNPS.

G. CHANGE 7: USER SUPPLIED TITLE

The user supplied title would not print on the graphics display produced with the command, PLHX. PLHX produces a display of a heat transfer analysis mesh. When the command, TITLE,N1, was issued, it seemed to be completely ignored and no user input was accepted. It was found that the PLOT10 subroutine TINSTR did not exist in the current PLOT10 library. TINSTR is used to accept alphanumeric input from the terminal and store it in an ASCII decimal equivalent array for later output by either of the PLOT10 subroutines, ANCHO or ANSTR. ANCHO allows the user to output a single alphanumeric character and ANSTR allows the user to output an alphanumeric string [Ref. 4: pp. 4-1,7-3,4].

TINSTR (written by David Marco of the NPS Naval Engineering Department) was added to the current PLOT10 library. This resulted in an output of the user supplied title, however, it was not printed in the same graphics window as the mesh display. The calls for subroutines, SCRDAT, USRTIT, and TITLE were moved to occur prior to the drawing of the mesh. SCRDAT prints the mesh dimensions, USRTIT prints the user supplied title, and TITLE prints the mesh analysis type information (heat transfer or structural). This resulted in the mesh, the user supplied title, and the mesh type description being displayed on the same screen with proper scaling of the mesh and no overlap. The bounds of the screen window also had to be changed to include all of the above mentioned items and the boxes drawn around them. This change was made in the subroutine BOX.

H. CHANGE 8: PRINTED OUTPUT

The graphic display from the command PLHX was unprintable using the print-screen option on the VAX computer. The borders of the program defined screen were too close to the borders of the PLOT10 window boundaries. This caused the printer to print "garbage." This was remedied by redefining the window a safe distance from the PLOT10 window boundaries. This required changing the arguments of the TWINDO (PLOT10 subroutine) call statement in the subroutine, BOX. It also required moving the coordinates of the user supplied title and of the mesh type description to conform to the new screen coordinates.

The above changes allowed the mesh and title information to be printed properly, however, the minimum and maximum values of the coordinate axes still did not print at all on the graphics output screen. Analysis of the PLOT10 subroutine, ANMODE, which is called by SCRDAT to print this data, revealed that the file read by ANMODE was not the same as the file being written to by SCRDAT. The file number in SCRDAT was changed to match the file read by ANMODE. Also the character size was changed to insure that the data remained inside the new screen and box coordinates. This allowed the entire graphics display to print properly.

I. CHANGE 9: ODE COMMAND

The ODE command solves first order ordinary differential equations for heat transfer systems. The instructions for use of the ODE command were incomplete. The "HELP" file failed to explain to the user how to input the initial nodal temperatures. After analyzing the appropriate subroutines for the required input data, the "HELP" file and user's manual were updated to include the necessary instructions.

J. CHANGE 10: STRUCTURAL GRAPHICS

This change is similar to change seven for the heat transfer mesh plotting subroutine, FPPLOT. The subroutine, CLPLOT, plots the structural analysis mesh. The calls for subroutines, SCRDAT, USRTIT, and TITLE were moved to occur prior to the drawing of the mesh to allow the mesh, user supplied title, and screen data to be displayed on the same graphics screen.

K. CHANGE 11: THREE DIMENSIONAL HEAT TRANSFER MESH

This change is similar to changes seven and ten but applies to the subroutine, FPPLOT. This subroutine plots the mesh for three dimensional heat transfer problems. The calls for subroutines, SCRDAT, USRTIT, and TITLE were moved to occur prior to drawing the mesh.

L. CHANGE 12: SMALL GRAPHICS OPTION

The only graphics output that could be printed was a full size output. This change was implemented to allow the user the option for a smaller output that could easily be included in a report or a thesis. The smaller output is 9 inches by 6.25 inches when printed.

This addition required changes to subroutines, GRAFI, FPPLOT, FP3PLT, CLPLOT, BOX, SCRDAT, TITLE, and USRTIT, all of which are located in the Group 5 set of subroutines. The variable, ISMALL, was added to the calls for each one. A user prompt was added to query the user to chose either the small or large graphics option.

To execute the small graphics option the user only has to respond "yes" to the prompt that occurs after issuing the command, GRAPH. The output will be scaled down without any other operator actions other than the normal graphics commands (i.e., PLHX or PLST). The output can be printed with the VAX print-screen option or with the hardcopy option added by Change 15.

This change does not apply to the IBM version operations.

M. CHANGE 13: CUBIC COMMAND

This change adds the capability to solve for the roots of a cubic equation. The coefficients or the roots can be real or complex. The cubic command was added in the Group One section of CALNPS. The output is written to a file in the user's local directory. The user must supply the name of the file at the prompt. The solution is automatically refined using the Newton-Raphson iteration method.

N. CHANGE 14: XYPLOT COMMAND

This change adds the capability of plotting curves from within the CALNPS domain. Up to two curves may be plotted on the same graph. The data is read from a data file previously prepared. A hardcopy of the plots may be obtained by responding to the appropriate prompts.

The original XY PLOTTER program was written by David Marco. It was modified by breaking it into two sets of subroutines and adding them to the CALNPS code in Group Five. The first set plots the curves on the screen using the PLOT10 graphics library. The second set creates files for hardcopy printing (see Change 15).

O. CHANGE 15: GRAPHICS HARDCOPY OPTIONS

This change adds the capability to print graphics by creating and printing files rather than using the print-screen option on the VAX computer. The code change discussed previously in Change 14 was expanded to work for all the graphics capabilities of CALNPS. The structural or heat transfer analysis meshes are printed on the screen using the PLOT10 graphics library and the hardcopy files are created using the DISSPLA graphics library. Files can be created for use on either of the printers currently available (LA75 or LA210). High resolution graphics is available for the LA75 printer and provides an extremely high quality printout. Execution of the hardcopy options is controlled entirely with user prompts.

P. CHANGE 16: DEMONSTRATION FACILITY

This change adds the capability for the user to select a demonstration file from a menu and run the problem from within the CALNPS domain. A new command, DEMO, was added to achieve this purpose. DEMO causes a menu of available sample problem files to be printed on the screen. The user only has to enter the name of the file desired. The current files are complete with graphics displays. It is assumed here that the user is familiar with the VAX computer system and knows how to manipulate the graphics screens with the mouse.

Q. CHANGE 17: MEMFRC COMMAND

The MEMFRC command calculates the member forces in a two dimensional structural system. When the MEMFRC command was executed, a FORTRAN error occurred indicating an access violation. Troubleshooting led to the discovery that the address of the integer array containing the row numbers of the displacement matrix which were to be multiplied by the stiffness matrix was incorrect. The variables N2 and

N3 in the subroutine, MEMFRC, were discovered to be interchanged. After correcting this error the forces were calculated correctly with no occurrence of a FORTRAN error.

IV. CALNPS OPERATIONS

This chapter describes in detail the procedure for using each command of CALNPS. Example problems including input and output data files are also included. This chapter is geared towards the user while the rest of this work is more geared towards modifying and maintaining the program. This chapter is a revision of the material originally published by Professor Wilson as Reference 1. Additions have been made as they apply to this version of CALNPS. A "+" symbol above the matrix designator indicates the formation of a new matrix. A matrix previously defined with the same name will be deleted. A "-" indicates that an existing matrix will be modified. The theory behind the numerical methods used in CALNPS is available in the current version of the user's manual as a reprint of Professor Wilson's class notes. The author wishes to express appreciation to Professor Wilson for permission to use this material.

A. GENERAL OPERATIONS

1. Description

As discussed in Chapter II, the commands that are classified as general commands perform standard functions that allow the user to interface with and control CALNPS.

2. Command Specifications

__CALNPS__

This command executes the CALNPS program on the VAX computer system. If issued within the "HELP" facility, a list of all CALNPS commands will be provided.

__DEMO__

This operation generates a menu of sample problem data files available to the user. Enter the name of the data file desired. The problem will be run entirely including graphics displays. After the graphics display is completed, type "CONTINUE" to return to CALNPS. The option for a hardcopy of the graphics display will always occur after the command, CONTINUE.

__HELP__

This operation enables the interactive "HELP" facility. The commands are filed alphabetically. Therefore, if several different commands are being requested, they will be found faster if requested in alphabetical order.

LABEL.N1

This operation will read and print N1 comment lines which follow the operation line. Column 1 of each line will be interpreted as a standard carriage control symbol (i.e., 0 for double space and 1 for skip to the top of the next page).

LIST

The list operation prints directory information for all arrays currently in storage and lists the amount of storage used.

NO

This operation suppresses all printing, except diagnostics, until the operation YES is encountered. This allows the user to suppress printing of data which has been proved to be correct in previous runs of CALNPS (see YES).

READ.N1

This operation permits the selection of a user's file or the terminal as the input file device. The file can be read using the file name or an assigned logical number.

N1 = XX	Subsequent commands will be read from file FOR0XX.DAT of the user's work space. The file can be named FOR0XX.DAT or be assigned a logical number with the VAX command, ASSIGN (i.e., ASSIGN FN.FT FOR0XX.DAT).
N1 = 5	Restores the terminal as the input file device. All files prepared for use with this command should end with either STOP or READ.5.

A prompt for the file name always follows the execution of the READ command. If the file is to be called by name, then N1 should be omitted. If a logical number is to be used, then press return at the prompt for a name.

RESUME.N1

This operation reads a saved file into memory. Any arrays currently in storage will be destroyed. The file must have been previously created in the user's directory using the SAVE operation. The default for N1 is N1 = 1 and it will be assumed that a structural problem is being resumed. The user will be prompted to enter the name of the file.

N1 = 1	A structural problem is being resumed.
N1 = 2	A heat transfer problem is being resumed.

SAVE,N1

This operation saves all arrays in storage at the time of issuance. Saved arrays will contain all modifications made since their creation. The default is N1 = 1 and it will be assumed that the problem being saved is a structural problem. The user will be prompted to enter a name consisting of a maximum of eight characters to be assigned to the file.

N1 = 1	A structural problem is being saved.
N1 = 2	A heat transfer problem is being saved.

START

This operation eliminates all arrays which were previously loaded or generated.

STOP

This operation causes normal termination of CALNPS program operations.

WRITE,N1

This operation will produce a file in the user's directory. A prompt appears after the WRITE command is encountered by CALNPS requesting a name to be assigned to the output file. All output will be directed to the output file until a WRITE,6 command is executed. An entry for N1 is required. Failure to make an entry for N1 will result in the erasure of any files read during the operation.

N1 = 6	Restores the terminal as the output file device.

YES

This operation negates the NO operation and permits all printing to resume to the designated file (See NO).

B. GENERAL MATRIX OPERATIONS

1. Description

CALNPS has the capability to perform most of the standard matrix manipulation operations. In addition there are several nonstandard array operations which are useful in engineering analysis.

2. Command Specifications

ADD,M1,M2

This operation will replace matrix M1 with the sum of matrices M1 and M2.

ADDSM,$\bar{M}$1,M2,N1,N2

This operation adds the smaller matrix M2 to the larger matrix M1 starting at the row number N1 and column number N2 of M1. This operation cannot enlarge the matrix M1.

COSEL,$\bar{M}$1

This operation calculates the cosine of each element in matrix M1. The elements of M1 must be expressed in radians.

CUBIC

This operation solves for the roots of a third order equation. The coefficients or the roots or both may be either real or complex. The required data entries are clearly explained by the user prompts. Complex coefficients are entered in the following format:

(Real part, Imaginary part).

DELETE,$\bar{M}$1

This operation will delete the array named M1 from storage.

DUP,M1,$\overset{+}{M}$2

This operation will form an array named M2 which is identical to the array named M1.

DUPDG,M1,$\overset{+}{M}$2

This operation forms a new row matrix named M2 from the diagonal terms of matrix M1. This operation is valid for real matrices only.

DUPSM,M1,$\overset{+}{M}$2,N1,N2,N3,N4

This operation forms a new submatrix named M2 with N3 rows and N4 columns from terms within the matrix named M1. The first term of matrix M2, M2(1,1), will be from row N1 and column N2 of matrix M1, M1(N1,N2). This operation is valid for real matrices only.

INVEL,M1

This operation replaces each term in the matrix named M1 with its reciprocal.

$^{+}$
LOAD,M1,N1,N2,N3

This operation will load an array of real numbers named M1 which has N1 rows and N2 columns. The terms of the array are entered in row-wise sequence on data lines following this operation. N3 is optional and defaults to free format. Integer arrays must be entered with the LOADI command.

N3 = 0	The data should be entered in a format of (8F10.0).
N3 = 1	An additional line, which contains the format of the data lines, precedes the data lines. For example, if the data is to be four numbers per line in field widths of 15, the additional line would contain the following information: (4F15.0). Note! Parenthesis are required.
N3 = 9	The data lines will be read in free format.

$^{+}$
LOADI,M1,M2,N1,N2,N3,N4

This operation will load an integer array named M1 which has N1 rows and N2 columns. The terms of the array are written in row-wise sequence on data lines which follow the operation command N3 is optional and defaults to free format.

N3 = 0	An additional line containing the format of the data lines must follow this operation command and precede the data. For example, if the data is to be four numbers per line in field widths of 10, the additional line would contain the following information: (4I10). Note! Parenthesis are required.
N3 = 9	Data will be read in free format.
N4	N4 is optional. The matrix M1 will be printed in partitioned form with N4 columns per partition. Lines have (N4 + 1)*5 characters. N4 defaults to 20 causing 125 characters to print per line.
M2	M2 is optional. If the letters "INCR" are entered as M2, there is a generation capability. Data must be entered as follows:

ITEM	CONTENTS
1	Row number.
2	Value 1.
3	Value 2.
Etc.	
N2 + 1	Value N.
N2 + 2	Generation code.

If the generation code is not zero, then the next line must contain the following data:

ITEM	CONTENTS
1	Row number increment.
2	Value 1 increment.
3	Value 2 increment.
Etc.	

N2+1	Value N increment.
N2+2	Last row to be generated.

This operation must be terminated by a row of at least N2+2 zeros, separated by commas or spaces. The user must ensure that each element is defined.

LOG,$\overset{-}{\text{M}}$1

This operation replaces each term in the matrix M1 with the natural log of the term.

MAX,M1,$\overset{+}{\text{M}}$2

This operation forms a column matrix named M2 in which each row contains the maximum absolute value of the corresponding row in matrix M1. The maximum and its column number are printed for each row. This operation is valid for real matrices only.

MULT,M1,M2,$\overset{+}{\text{M}}$3

This operation generates a new matrix M3 which is the product of matrices M1 and M2 (M3 = M1*M2). This operation is valid for real arrays only.

NORM,M1,$\overset{+}{\text{M}}$2,N1

If N1 equals zero, a row matrix named M2 is formed in which each column contains the sum of the absolute values of the corresponding column of matrix M1. If M1 does not equal zero, a row matrix named M2 is formed in which each column contains the square root of the sum of the squares of the values of the corresponding columns of the matrix M1.

PRINT,M1,N1,N2,N3

This operation will print the array named M1 in a matrix format of up to eight columns per line. N1, N2, and N3 are optional.

N1	N1 comment lines, which follow the operation line, will be read and printed. N1 defaults to zero.
N2	The matrix will be printed in partitioned form with N2 columns per partition. Lines will have N2*15 + 5 characters. N2 defaults to eight, printing 125 characters per line.

N3	If N3 is greater than zero, integer format (I6) is used. The array must be an integer array that was previously loaded with the LOADI command. The default value is zero and real format (F15.7) is used.

PROD,M1,$\overset{+}{\text{M}}$2

This operation forms a 1 X 2 array named M2 which contains the product of all terms in the matrix named M1. The product, X, is stored as two numbers of the form: X = P*10**E in which M2(1) = P and M2(2) = E, the exponent.

SCALE,$\bar{\text{M}}$1,M2

This operation replaces each term in the matrix named M1 with the term multiplied by the term M2(1,1) of the matrix named M2. This operation is valid for real matrices only.

SINEL,$\bar{\text{M}}$1

This operation will calculate the sine of each element of matrix M1. The elements of M1 must be expressed in radians.

SOLVE,$\bar{\text{M}}$1, $\bar{\text{M}}$2,N1,N2

This operation solves the matrix equation, $[A][X] = [B]$ where M1 is the name of the A matrix and M2 is the name of the B matrix, unless modified as indicated below:

N1 = 0	The system is solved using a compact forward backward substitution process. M1 is replaced by the compact L U form, where L is a lower unit matrix. The signs of the off diagonal terms are changed for convenience purposes. The values of the matrix X are stored in the M2 matrix.
N1 = 1	Matrix M1 is transformed into the compact L U form only. No equation solving is done and M2 is not changed at all.
N1 = 2	For a given matrix M2 and the matrix M1 previously transformed, matrix M2 is replaced by the values of the matrix X. Forward backward substitution is used.
N1 = 3	Matrix M1 is replaced by its inverse for symmetric matrices only.
N2 = 0	Matrix M1 is symmetric. N2 defaults to zero.
N2 = 1	For symmetric matrices, matrix M1 is factored into the LDL form. The diagonal matrix D is stored on the diagonal of M1. The parameter N2 permits the direct solution of nonsymmetric systems of equations. If N2 is not equal to zero, an L U decomposition of matrix M1 will be performed.

SQREL,$\bar{\text{M}}$1

This operation replaces each term in the matrix named M1 with the square root of the term.

STODG,$\bar{\text{M}}$1,M2

This operation stores a row or column matrix named M2 at the diagonal locations of matrix M1. This operation is valid for real matrices only.

STOSM,$\bar{\text{M}}$1,M2,N1,N2

This operation stores a submatrix named M2 within the existing matrix M1. The first term of the submatrix M2 will be stored at row N1 and column N2 of matrix M1. The terms within the area of M1 in which M2 is stored will be destroyed.

SUB,$\bar{\text{M}}$1,M2

This operation will replace matrix M1 with matrix M1 less matrix M2. This operation is valid for real matrices only.

SUBSM,$\bar{\text{M}}$1,M2,N1,N2

This operation subtracts matrix M2 from the larger matrix M1 starting at row number N1 and column number N2 of the larger matrix.

TRAN,M1,$\overset{+}{\text{M}}$2

This operation generates a new matrix M2 which is the transpose of matrix M1. This operation is valid for real matrices only.

ZERO,$\overset{+}{\text{M}}$1,N1,N2,N3,N4

A real matrix named M1 is created with N1 rows and N2 columns. The terms in this matrix will have the following values:

$$\text{M1(I,J)} = \text{N3} \quad \text{I} = 1,\ldots,\text{N1}$$
$$\text{M1(I,J)} = \text{N4} \quad \text{J} = 1,\ldots,\text{N2}$$

N3 and N4 are optional and if left blank, M1 will be a null matrix. If N3 and N4 are both = 1, M1 will be a unit matrix.

3. Examples

a. *Use of LOADI command*

This example demonstrates the use of the LOADI command with the generation option. The example shows the generation of a 6 X 6 integer array.

(1) Input Data File. The file demonstrates the use of the START, WRITE, LABEL, and READ commands as well as the LOADI command. The WRITE,1 command on line two causes the output data to be printed to a data file in the user's directory. The LABEL command allows the input of comment statements. The first data line after the LOADI command defines the values of the first row of the array. The second data line causes rows three and five to be generated by incrementing row one by the given values. The third data line defines the second row of the array. The fourth data line causes row four and six to be generated by incrementing row two by the given values. The final data line must be a row of at least N+2 zeros separated by commas or spaces. N represents the number of columns in the array.

```
START
WRITE,1
LABEL,1
    GENERATED INTEGER ARRAY EXAMPLE
LOADI,M1,INCR,6,6,9
1,1,1,1,1,1,1,1
2,1,1,1,1,1,1,5
2,2,2,2,2,2,2,1
2,2,2,2,2,2,2,6
0,0,0,0,0,0,0,0
WRITE,6
READ,5
```

(2) Output Data File.

```
LABEL,1
    GENERATED INTEGER ARRAY EXAMPLE
LOADI,M1,INCR,6,6,9
 6 ROWS    6 COLUMNS
     1   2   3   4   5   6
1    1   1   1   1   1   1
2    2   2   2   2   2   2
3    2   2   2   2   2   2
4    4   4   4   4   4   4
5    3   3   3   3   3   3
6    6   6   6   6   6   6

WRITE,6
```

b. Use of SOLVE command

This example demonstrates the different uses of the SOLVE command. Comment lines are interspersed throughout the data file to explain how the SOLVE command is being used.

(1) Input Data File.

```
START
WRITE,1
LOAD,M1,3,3,9
1 1 2
2 4 -3
3 6 -5
LOAD,M2,3,1,9
9 1 0
PRINT,M1,7,3
*** SOLVE A SYSTEM OF LINEAR EQUATIONS
*** MATRIX EQUATION AX=B
   X +  Y + 2Z = 9
  2X + 4Y - 3Z = 1
  3X + 6Y - 5Z = 0
   M1=A    M2=B
************************************************
PRINT,M2
LABEL,2
 ** REPRODUCE M1 AND M2 FOR USE
 ** IN LATER PART OF EXAMPLE
DUP,M1,M3
DUP,M2,M4
SOLVE,M1,M2,0,1
LABEL,8
*****************************************************************
* N1=0 WHICH CAUSES M1 TO BE TRIANGULARIZED.
* M1 IS STORED IN A COMPACT L/U FORM, WHERE L IS A LOWER
* UNIT MATRIX WITH THE SIGNS CHANGED FOR CONVENIENCE
* PURPOSES. THE VALUES OF THE X MATRIX (SOLUTION) WILL
* BE STORED IN THE M2 MATRIX.
* N2=1 INDICATES THAT M1 IS UNSYMMETRIC.
* ***************************************************************
PRINT,M1
PRINT,M2,1
        ****** SOLUTION ******
PRINT,M3
PRINT,M4
SOLVE,M3,M4,1,1
LABEL,2
** N1=1 CAUSES M3 TO BE TRIANGULARIZED AS BEFORE.
** M4 WILL BE UNCHANGED.
SOLVE,M3,M4,2,1
LABEL,3
** N1=2 CAUSES THE MATRIX M3 TRIANGULARIZED IN THE
** PREVIOUS STEP TO BE SOLVED WITH M4.  M4 IS
** REPLACED WITH THE VALUES OF THE X MATRIX.
PRINT,M3
PRINT,M4
LOAD,M1,2,2,9    ** EXAMPLE OF INVERSE OPERATION **
3 2 2 2
PRINT,M1
LOAD,M2,2,1,9
1 1
PRINT,M2
SOLVE,M1,M2,3,0
LABEL,2
** THIS OPERATION WILL COMPUTE THE INVERSE OF A
** SYMMETRIC MATRIX.  NO EQUATION SOLVING IS DONE.
PRINT,M1,1
** INVERSE MATRIX **
LABEL,5
*******************************************************************
* EXAMPLE OF SOLVING A SYSTEM OF LINEAR EQUATIONS
* WHERE THE COEFFICIENT MATRIX IS SYMMETRIC.
* M5 IS DIAGONALIZED AND THE DIAGONAL MATRIX
* M6 IS STORED ON THE DIAGONAL OF M5.
*******************************************************************
LOAD,M5,3,3,9
1 2 0
2 2 1
0 1 3
LOAD,M6,3,1,9
1 5 3
PRINT,M5
PRINT,M6
SOLVE,M5,M6,0,0
PRINT,M5
PRINT,M6,1
  *** SOLUTION MATRIX ***
WRITE,6
READ,5
```

(2) Output Data File.

```
LOAD,M1,3,3,9
 3 ROWS    3 COLUMNS
LOAD,M2,3,1,9
 3 ROWS    1 COLUMNS
PRINT,M1,6,3
** SOLVE A SYSTEM OF LINEAR EQUATIONS
*** MATRIX EQUATION AX=B
   X +  Y + 2Z = 9
  2X + 4Y - 3Z = 1
  3X + 6Y - 5Z = 0
  M1=A    M2=B
*************************
```

```
          1                2                 3
  1  1.0000000D+00  1.0000000D+00   2.0000000D+00
  2  2.0000000D+00  4.0000000D+00  -3.0000000D+00
  3  3.0000000D+00  6.0000000D+00  -5.0000000D+00
PRINT,M2
          1
  1  9.0000000D+00
  2  1.0000000D+00
  3  0.0000000D+00
LABEL,2
 ** REPRODUCE M1 AND M2 FOR USE
 ** IN LATER PART OF EXAMPLE
DUP,M1,M3
 3 ROWS    3 COLUMNS
DUP,M2,M4
 3 ROWS    1 COLUMNS
SOLVE,M1,M2,0,1
LABEL,S
*************************************************************
* N1=0 WHICH CAUSES M1 TO BE TRIANGULARIZED.
* M1 IS STORED IN A COMPACT L U FORM, WHERE L IS A LOWER
* UNIT MATRIX WITH THE SIGNS CHANGED FOR CONVENIENCE
* PURPOSES.  THE VALUES THE X MATRIX (SOLUTION) WILL
* BE STORED IN THE M2 MATRIX.
* N2=1 INDICATES THAT M1 IS UNSYMMETRIC.
*************************************************************
PRINT,M1
          1                2                 3
 1  1.0000000D+00   1.0000000D+00   2.0000000D+00
 2 -2.0000000D+00   2.0000000D+00  -7.0000000D+00
 3 -3.0000000D+00  -1.5000000D+00  -5.0000000D+00
PRINT,M2,1
    ****** SOLUTION ******
          1
 1  1.0000000D+00
 2  2.0000000D+00
 3  3.0000000D+00
PRINT,M3
          1                2                 3
 1  1.0000000D+00   1.0000000D+00   2.0000000D+00
 2  2.0000000D+00   4.0000000D+00  -3.0000000D+00
 3  3.0000000D+00   6.0000000D+00  -5.0000000D+00
PRINT,M4
          1
 1  9.0000000D+00
 2  1.0000000D+00
 3  0.0000000D+00
SOLVE,M3,M4,1,1
TRIANGULARIZE ONLY
MATRIX M3 IS TRANSFORMED INTO A COMPACT L U FORM WHERE
L IS A LOWER UNIT MATRIX WITH SIGNS OF OFF DIAGONAL
TERMS CHANGED AND U IS THE UPPER.
```

```
LABEL,2
** N1 = 1 CAUSES M3 TO BE TRIANGULARIZED AS BEFORE.
** M4 WILL BE UNCHANGED.
SOLVE,M3,M4,2,1
FORWARD REDUCTION AND BACK SUBSTITUTION
SOLUTION STORED IN MATRIX M2
LABEL,3
** N1 = 2 CAUSES THE MATRIX M3 TRIANGULARIZED IN THE
** PREVIOUS STEP TO BE SOLVED WITH M4.  M4 IS
** REPLACED WITH THE VALUES OF TH X MATRIX.
PRINT,M3
             1                 2                  3
 1  1.0000000D+00   1.0000000D+00   2.0000000D+00
 2 -2.0000000D+00   2.0000000D+00  -7.0000000D+00
 3 -3.0000000D+00  -1.5000000D+00  -5.0000000D-01
PRINT,M4
             1
 1  1.0000000D+00
 2  2.0000000D+00
 3  3.0000000D+00
LOAD,M1,2,2,9  ** EXAMPLE OF INVERSE OPERATION **
ARRAY M1      DELETED
 2 ROWS    2 COLUMNS
PRINT,M1
             1                 2
 1  3.0000000D+00   2.0000000D+00
 2  2.0000000D+00   2.0000000D+00
LOAD,M2,2,1,9
ARRAY M2        DELETED
 2 ROWS    1 COLUMNS
PRINT,M2
             1
 1  1.0000000D+00
 2  1.0000000D+00
SOLVE,M1,M2,3,0
MATRIX INVERSION ONLY
LABEL,2
** THIS OPERATION WILL COMPUTE THE INVERSE OF A
** SYMMETRIC MATRIX.  NO EQUATION SOLVING IS DONE.
PRINT,M1,1
** INVERSE MATRIX **
             1                 2
 1  1.0000000D+00  -1.0000000D+00
 2 -1.0000000D+00   1.5000000D+00
LABEL,5
*******************************************************
* EXAMPLE OF SOLVING A SYSTEM OF LINEAR EQUATIONS
* WHERE THE COEFFICIENT MATRIX IS SYMMETRIC.
* M5 IS DIAGONALIZED AND THE DIAGONAL MATRIX
* IS STORED ON THE DIAGONAL OF M5.
*******************************************************
LOAD,M5,3,3,9
```

```
 3 ROWS    3 COLUMNS
LOAD,M6,3,1,9
 3 ROWS    1 COLUMNS
PRINT,M5
            1                2                3
 1  1.0000000D+00   2.0000000D+00   0.0000000D+00
 2  2.0000000D+00   2.0000000D+00   1.0000000D+00
 3  0.0000000D+00   1.0000000D+00   3.0000000D+00
PRINT,M6
            1
 1  1.0000000D+00
 2  5.0000000D+00
 3  3.0000000D+00
SOLVE,M5,M6,0,0
PRINT,M5
            1                2                3
 1  1.0000000D+00   2.0000000D+00   0.0000000D+00
 2  2.0000000D+00  -2.0000000D+00  -5.0000000D-01
 3  0.0000000D+00   1.0000000D+00   3.5000000D+00
PRINT,M6,1
 *** SOLUTION MATRIX ***
            1
 1  2.7142857D+00
 2 -8.5714286D-01
 3  1.2857143D+00
WRITE,6
```

C. STATIC ANALYSIS OPERATIONS

1. Description

The commands in this section serve to solve static structural analysis problems. These operations form the total stiffness and diagonal (lumped) mass matrices for systems of two or three dimensional elements.

For a two dimensional frame problem, the FRAME, ADDK, LOADI, SOLVE, and MEMFRC commands are used. The FRAME command forms the 6 X 6 element stiffness matrix and the 3 X 6 force-displacement matrix. An integer array relating the members to the equilibrium equation numbers must be generated using the LOADI command. The ADDK command is then used to add the element stiffness matrix to the total stiffness matrix. The system loads must be entered as a row array with the LOAD command. The displacements can then be solved for with the SOLVE command. Finally, the forces in each member can be computed by the MEMFRC command. The frame example at the end of this section clearly demonstrates this process.

Also for two dimensional analysis, the SLOPE command which forms a 4 X 4 stiffness matrix for a beam or column member from the classical slope deflection equations is available. The element stiffness, mass, and stress-displacement transformation matrices for two dimensional three to eight node isoparametric elements can be calculated with the PLANE command.

For a three dimensional analysis of a beam or a truss, the commands BEAM (or) TRUSS should be used in conjunction with the NODES, BOUND, ADDSF, LOADS, SOLVE, DISPL, FORCE, and LOADI commands. The NODES command is used first to define the geometry of the system by generating an array of the coordinates of the joints in the system. The BOUND command specifies the nonzero displacements of the joints in the system. Each joint may have from zero to six displacement degrees of freedom. Material properties are loaded in an array in accordance with the instructions for the BEAM and TRUSS commands. The BEAM or TRUSS commands are then used to calculate the element stiffness, mass, and force-displacement matrices. The next step is the direct addition of element stiffnesses to form the total stiffness and diagonal mass matrix of the system with the ADDSF command. Loads on the system are introduced by means of the LOADS command. The displacements can now be solved for with the SOLVE command and printed with the DISPL command. Member forces can be found with the FORCE command. If a two dimensional truss analysis is desired, simply enter zero for one of the dimension coordinates (NODE command).

2. **Command Specifications**

ADDK,$\bar{\text{M}}$1,M2,M3,N1

This operation adds the element stiffness matrix named M2 to the total stiffness matrix named M1, where M1 was previously defined and initially set to zero. M3 is the name of the integer array in which the column number N1 contains the row or column numbers in the total stiffness matrix where the element stiffness terms are to be added.

ADDSF,$\overset{+}{\text{M1}}$, $\overset{+}{\text{M2}}$

This operation forms the total stiffness matrix named M1 and a lumped mass matrix named M2 for the structural system from the element stiffness and mass matrices which are stored on low speed storage. These matrices can be printed with the PRINT command. If M2 is not specified, the row mass matrix M2 will not be formed.

BEAM,$\overset{+}{\text{M1}}$,M2,M3,M4

This operation calculates the element stiffness, mass and force-displacement transformation matrices for three dimensional beam members. These arrays are stored in sequence on low speed storage to be used by other operations.

M1	The name of the beam element group.
M2	The name of the coordinate array.
M3	The name of the boundary condition array.
M4	The name of an array which contains beam properties and has been loaded by the standard matrix LOAD command. One data line for each beam in this group of beam elements must follow this operation command. The beam data should be entered in free format as shown below:

ITEM	CONTENTS
1	Beam identification number.
2	Node number I.
3	Node number J.
4	Node number K.
5	Beam property number NP as explained below.

This sequence of data must be terminated by a row of at least five zeros separated by spaces or commas.

a. Property Data

The material and geometric properties for each element are specified in the M4 array in the following order:

M4(NP,1) = Axial area of member A.
M4(NP,2) = Torsional moment of inertia, J.
M4(NP,3) = Moment of inertia about axis 2, I (See Figure 2).
M4(NP,4) = Moment of inertia about axis 3, I (See Figure 2).
M4(NP,5) = Modulus of elasticity, E.
M4(NP,6) = Shear modulus, G.
M5(NP,7) = Mass per unit length of beam.

NP is the specific material property number specified as item five of the beam data above.

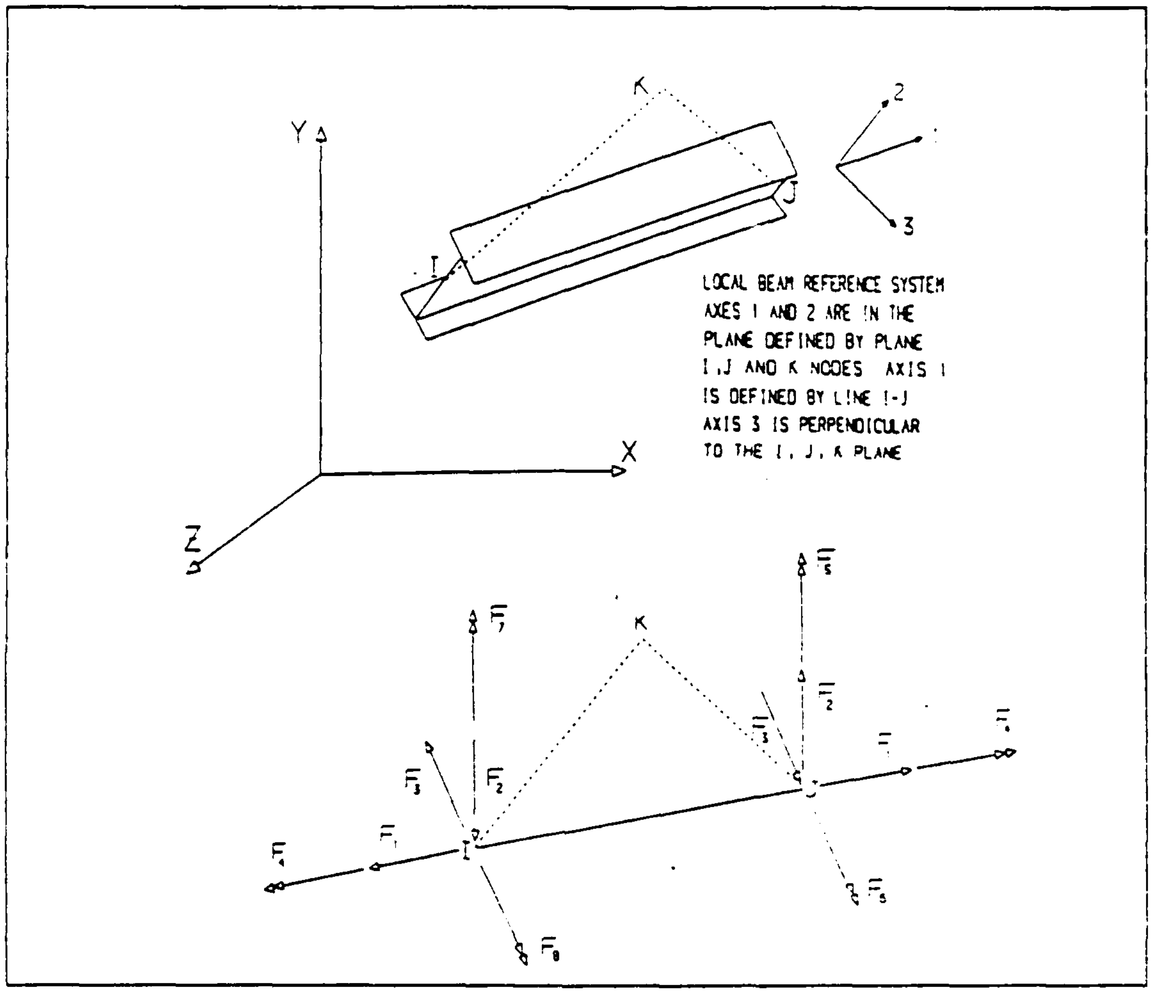


Figure 2. Local Sign Convention For BEAM Command.

BOUND,M1

This operation specifies the displacements which are nonzero for the structural system of joints specified by the NODES command.

M1	The name of the boundary condition array to be generated. This operation must be followed by a series of data lines containing the following information in free format:
ITEM	CONTENTS
1	The node number for the first node in a series of nodes with identical displacement specification.
2	The node number for the last node in the series.
3	X-Translation.

4	Y-Translation.
5	Z-Translation.
6	X-Rotation.
7	Y-Rotation.
8	Z-Rotation.
9	The node number increment used to generate conditions for additional nodes.

A translation or rotation equals: (A) Zero for zero or undefined displacements, or (B) one for nonzero displacements to be evaluated by other operations. If a node boundary condition is not specified, all displacements at that node are assumed zero. If boundary conditions are specified more than once, the last definition is used. It is not necessary to enter the data in any order of nodal numbers. This sequence of data must be terminated by a row of at least nine zeros separated by spaces or commas.

DISPL,M1,M2

This operation prints the displacement matrix named M1 in joint sequence order, where M2 is the name of the boundary condition array.

FORCE,M1,M2,M3

This operation calculates the member forces for a group of elements.

M1	The name of the element group.
M2	The displacement matrix.
M3	The name of the matrix in which the forces are stored in the order calculated. M3 is optional and if not specified, the element forces will be printed only and not retained in storage.

FRAME,M1, M2

This operation forms a 6 X 6 stiffness matrix named M1 and a 3 X 6 force-displacement matrix named M2 for the two dimensional frame member shown in Figure 3. The properties of the member are defined on one data line immediately following the FRAME command line. This line of data should be entered in free format and contain the following information:

ITEM	CONTENTS
1	Axial area, A.
2	Modulus of elasticity, E.

3	Moment of inertia, I.
4	X(I).
5	Y(I).
6	X(J).
7	Y(J).

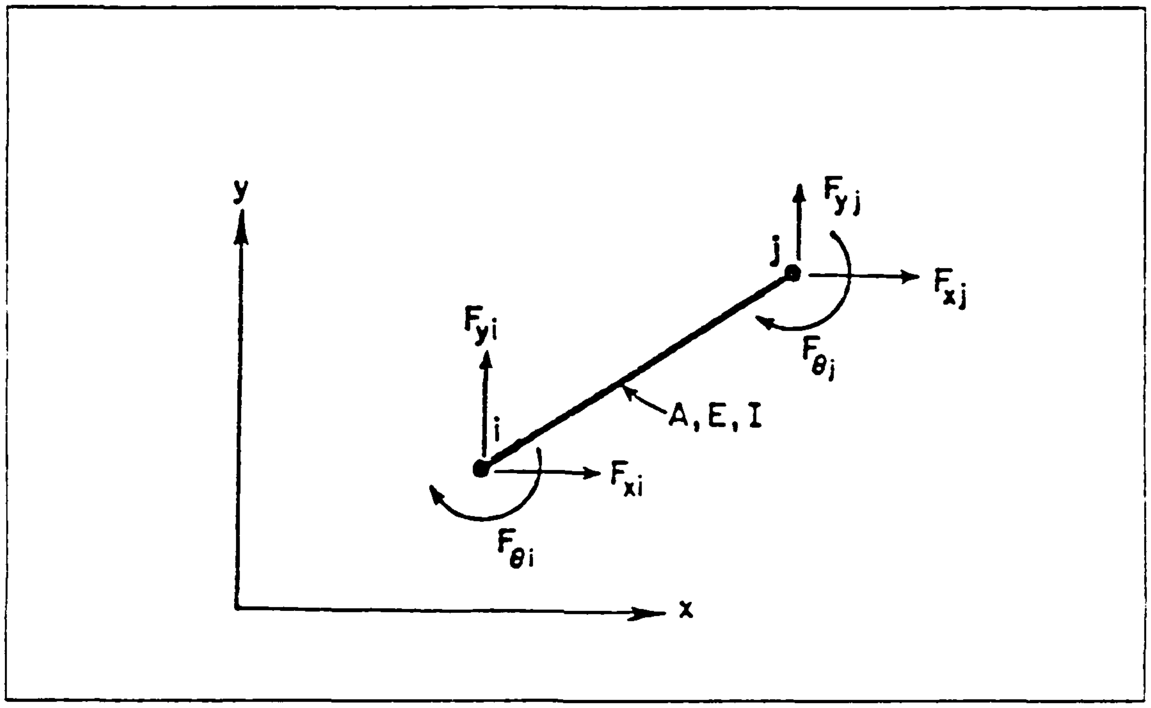


Figure 3. Two Dimensional Frame Member.

The positive definition of the element forces are shown in Figure 4 and the geometry and joint displacements are shown in Figure 5.

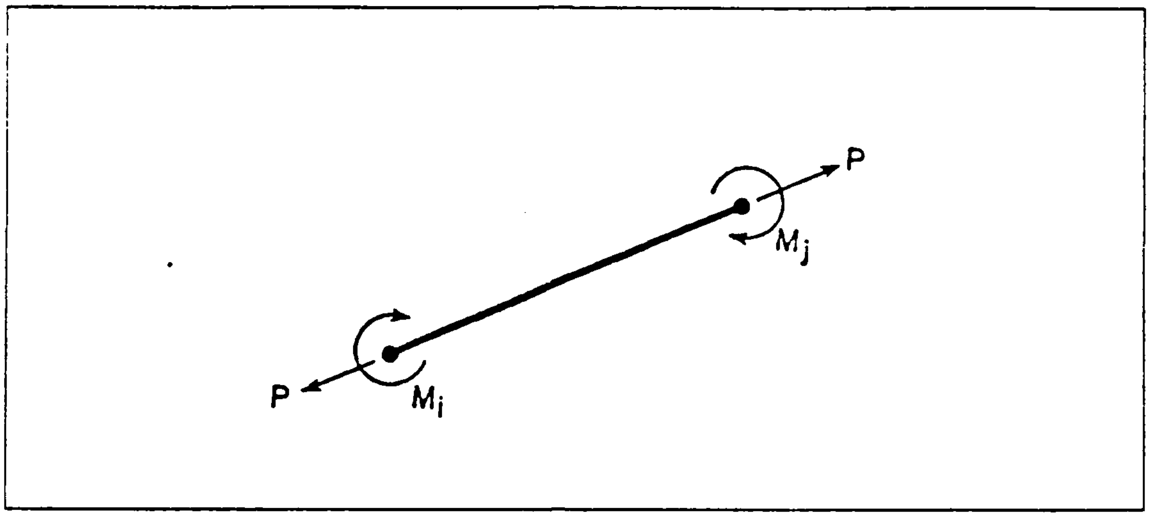


Figure 4. Positive Definition of Element Forces for FRAME Command.

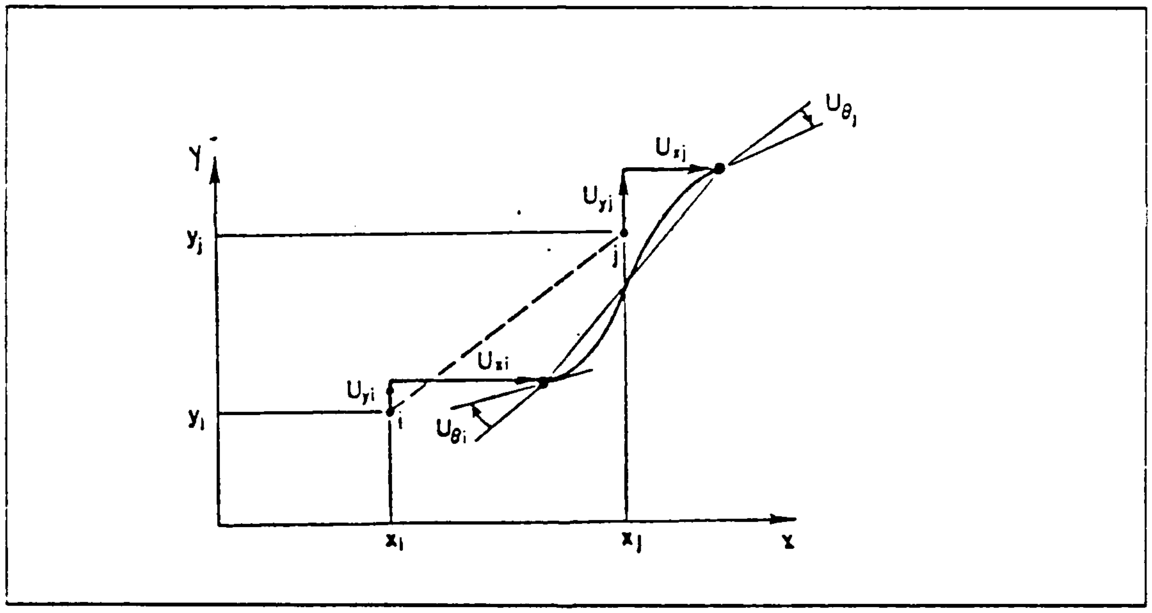


Figure 5. Geometry and Joint Displacements for Frame Command.

LOADS,M1,M2,N1

This operation forms a load matrix named M1 with N1 columns (N1 load conditions) where M2 is the name of the boundary condition array generated by the BOUND command. This operation must be followed by a series of data lines, one for each loaded joint for each load condition. The data lines must be entered in free format and contain the following information:

ITEM	CONTENTS
1	Joint number.
2	Load condition number.
3	Load in X-direction.
4	Load in Y-direction.
5	Load in Z-direction.
6	Moment about X-axis.
7	Moment about Y-axis.
8	Moment about Z-axis.

This sequence of data must be terminated by a row of at least eight zeros separated by spaces or commas.

MEMFRC,M1,M2,M3,$\overset{+}{\text{M}}$4,N1

This operation multiplies the element stiffness matrix named M1 by the joint displacement matrix named M2. M3 is the name of the integer array in which the column number N1 contains the row numbers in the displacement matrix M2, which are to be multiplied by the element stiffness (or force-displacement) matrix M1. The results of this multiplication are stored in the array named M4.

NODES,$\overset{-}{\text{M}}$1,N1,N2

This operation creates an array (N1,3) named M1 which contains the coordinates for all joints in a structural system. The following information must be entered in free format on data lines immediately following the NODES command:

ITEM	CONTENTS
1	Node number.
2	X-Coordinate
3	Y-Coordinate
4	Z-Coordinate.

If N2 is omitted or equal to zero, then no point generation is required and the input data is in cartesian coordinates. If either of these is not true, then N2 should be set equal to one. If N2 = 1, there is point generation and coordinate conversion capability and data should be entered in free format as follows:

ITEM	CONTENTS
1	Node number.
2	X-Coordinate.
3	Y-Coordinate.
4	Z-Coordinate.
5	System type.
6	Generation code.

System type refers to the system used when inputting the data. All coordinates will be converted to the Cartesian system for use by CALNPS. Positive coordinate systems are as shown in Figure 6. The system type codes are as follows:

SystemType	System
0,1	Cartesian.
2	Cylindrical, Z axis longitudinal.
3	Cylindrical, Y axis longitudinal.
4	Cylindrical, X axis longitudinal.
5	Spherical.

Note! The input data is the same as above with the following correspondence (angles are in degrees):

Cartesian	Cylindrical	Spherical
X	Rho	Rho
Y	Theta	Theta
Z	Z	Phi

If the generation code is not equal to zero, the next line is a generation vector for the self generation of nodes. It is formatted as follows:

ITEM	CONTENTS
1	Node number increment.
2	X increment.
3	Y increment.
4	Z increment.

5 Last node number to be generated.

It is assumed that the increments pertain to the same system of reference as the previous line. This operation must be terminated by a row of at least five zeroes separated by spaces or commas.

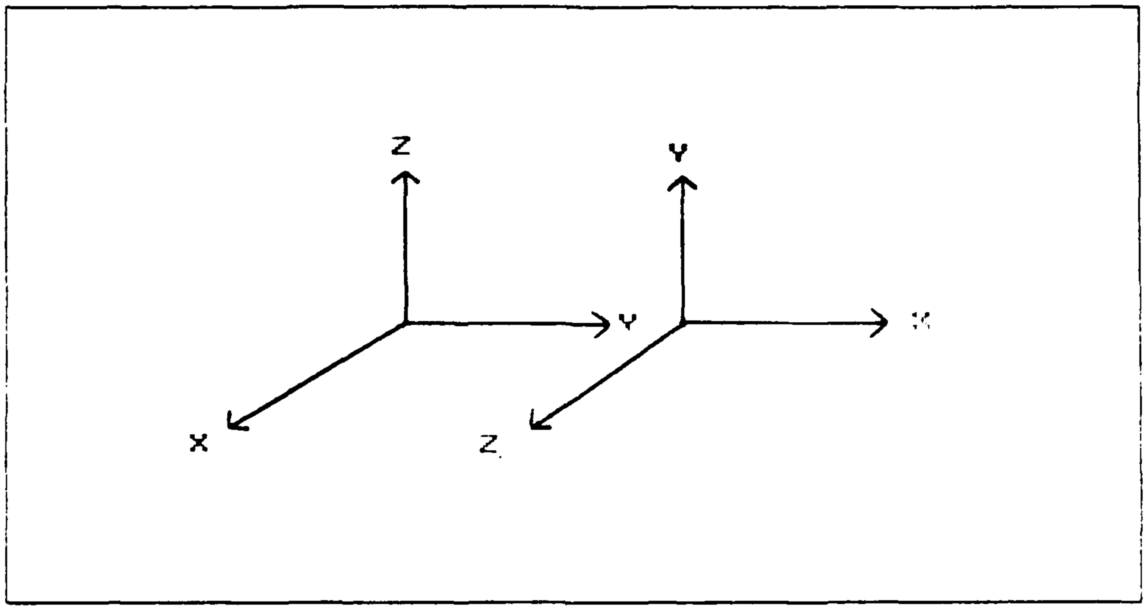


Figure 6. *Positive Coordinate Systems For NODES Command.*

PLANE,M1,M2,M3,M4,N1,N2

This operation calculates the element stiffness, mass and stress-displacement transformation matrices for three to eight node isoparametric elements. These arrays are stored in sequence as a group on low speed storage to be used later by other operations (ie. ADDSF and FORCE).

M1	The user defined name of the element group.
M2	The name of the joint coordinate array.
M3	The name of the boundary condition array.
M4	The name of the array which contains the material properties of the elements (one row per different material) where:

M4(NP,1) = Modulus of elasticity, E.
M4(NP,2) = Poisson's ratio, V.
M4(NP,3) = Thickness of element.
M4(NP,4) = Mass density of the element.
NP is the material identification number.

N1 Number of integration points in the R direction. (See Figure 7)

N2	Number of integration points in the S direction. (See Figure 7)

One line of data for each three to eight node element in the group must follow the PLANE operation line. The data should be entered in free format and should contain the following information:

ITEM	CONTENTS
1	Element identification number.
2	Node number N1.
3	Node number N2.
4	Node number N3.
5	Node number N4.
6	Node number N5.
7	Node number N6.
8	Node number N7.
9	Node number N8.
10	Material identification number NP.
11	Natural coordinate of stress output R1.
12	Natural coordinate of stress output S1.
13	Natural coordinate of stress output R2.
14	Natural coordinate of stress output S2.
15	Natural coordinate of stress output R3.
16	Natural coordinate of stress output S3.

N4 through N8 above are optional, however, zeros must be entered in their place if not used. The midside nodes, if present, must be within the center half of the side. The local numbering system for the element is demonstrated in Figure 7.

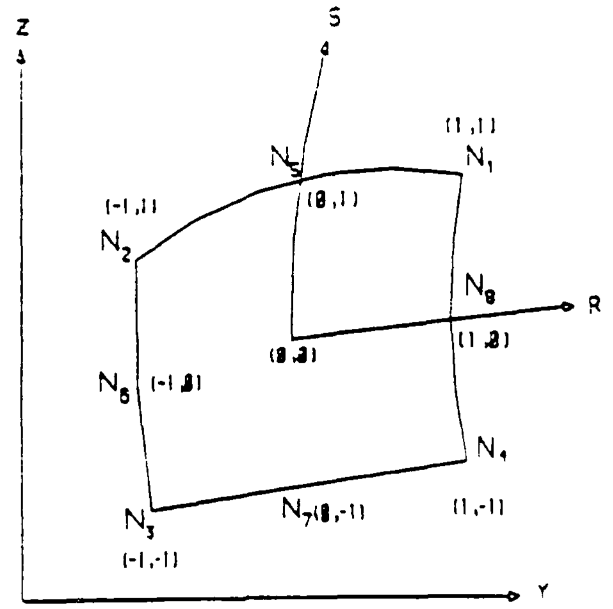


Figure 7. Numbering System for Isoparametric Elements (PLANE Command).

Stresses will be printed by the FORCE command at the three points defined by items 11-16 above. The forces are defined as follows:

$$\begin{bmatrix} F1 \\ F2 \\ F3 \\ F4 \\ F5 \\ F6 \\ F7 \\ F8 \\ F9 \end{bmatrix} = \begin{bmatrix} \sigma_{xx}(1) \\ \sigma_{yy}(1) \\ \tau_{xy}(1) \\ \sigma_{xx}(2) \\ \sigma_{yy}(2) \\ \tau_{xy}(2) \\ \sigma_{xx}(3) \\ \sigma_{yy}(3) \\ \tau_{xy}(3) \end{bmatrix}$$

SLOPE,M1

This operation forms a 4 X 4 stiffness matrix, M1 for a beam or column member from the classical slope deflection equations. The properties of the member are defined on one data line immediately following the command line. This data should be entered in free format and contain the following information:

ITEM	CONTENTS
1	Moment of inertia, I.
2	Modulus of elasticity, E.
3	Length of member, L.

The sign convention is defined as in Figure 8.

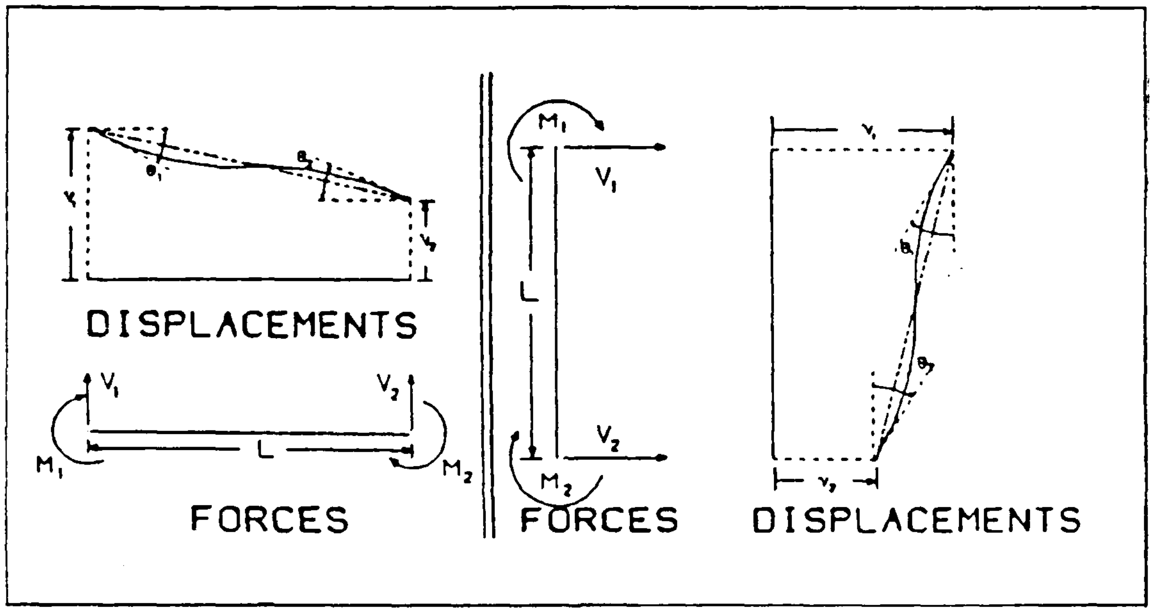


Figure 8. Sign Convention for SLOPE Command.

For this sign convention the classical slope deflection equations can be written as:

$$F_1 = \left(\frac{EI}{L}\right)\left[4V_1 + 2V_2 + 6\frac{(V_3 - V_4)}{L}\right]$$

$$F_2 = \left(\frac{EI}{L}\right)\left[2V_1 + 4V_2 + 6\frac{(V_3 - V_4)}{L}\right]$$

$$F_3 = -F_4 = \frac{(F_1 + F_2)}{L}$$

The forces can be found with the matrix equation, $F = KV$, where K is the 4 X 4 stiffness matrix formed by the SLOPE command.

TRUSS,M1,M2,M3,M4

This operation forms the element stiffness, mass, and force-displacement matrices for the three dimensional truss members (for a two dimensional analysis enter zero for all entries in one direction). The arrays are stored on low speed storage in sequence and will be used by other structural operations.

M1	The name of this group of truss members. M1 is generated by CALNPS.
M2	The name of the coordinate array.
M3	The name of the boundary condition array.
M4	An NP X 3 array of section properties in which NP is the number of different properties and:

M4(NP,1) = Cross sectional area, A.
M4(NP,2) = Modulus of elasticity, E.
M4(NP,3) = Mass per unit length of the member.
This matrix can be loaded using the LOAD command.

The TRUSS command must be followed immediately by one line of data per truss member. The data should be entered in free format and contain the following information:

ITEM	CONTENTS
1	Truss member identification number.
2	Joint number I.
3	Joint number J.
4	Section property number, NP.

This operation must be terminated by a row of at least four zeros separated by spaces or commas.

3. Examples

a. Two Dimensional Frame Member

This example was published by Professor Wilson in [Ref. 5: pp. 2.7-2.8]. The geometry of the six member frame is shown below in Figure 9. Due to the length of the output file, it is not presented here. This problem is available in the online demonstration facility.

MEMBER PROPERTIES

Members 1 to 4

$I = 1000 \text{ in}^4$

$A = 20 \text{ in}^2$

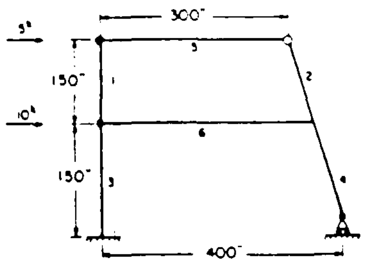
$E = 30{,}000$ ksi

Members 5 & 6

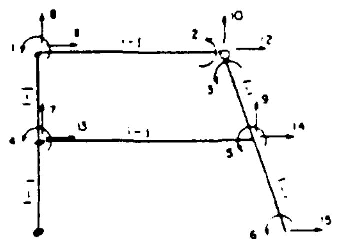
$I = 2000 \text{ in}^4$

$A = 30 \text{ in}^2$

$E = 30{,}000$ ksi



STRUCTURAL DIMENSIONS, PROPERTIES AND LOADS



NUMBERS ASSIGNED TO EXTERNAL LOADS "R" AND JOINT DISPLACEMENTS "U"

Member Force	Member 1	2	3	4	5	6
F1	13	14	0	15	11	13
F2	7	9	0	0	8	7
F3	4	5	0	6	1	4
F4	11	12	13	14	12	14
F5	8	10	7	9	10	9
F6	1	3	4	5	2	5

SUMMARY OF EQUILIBRIUM EQUATIONS

Figure 9. Direct Stiffness Method Example of Two Dimensional Frame.

(1) Input Data File.

```
START
FRAME,K1,T1
20 30000 1000 0 150 0 300
FRAME,K2,T2
20 30000 1000 350 150 300 300
FRAME,K3,T3
20 30000 1000 0 0 0 150
FRAME,K4,T4
```

```
20 30000 1000 400 0 350 150
FRAME,K5,T5
30 30000 2000 0 300 300 300
FRAME,K6,T6
30 30000 2000 0 150 350 150
LOADI,LM,6,6,9
13 14 0 15 11 13
7 9 0 0 8 7
4 5 0 6 1 4
11 12 13 14 12 14
8 10 7 9 10 9
1 3 4 5 2 5
ZERO,K,15,15
ADDK,K,K1,LM,1
ADDK,K,K2,LM,2
ADDK,K,K3,LM,3
ADDK,K,K4,LM,4
ADDK,K,K5,LM,5
ADDK,K,K6,LM,6
LOAD,R,15,1,9
0 0 0 0 0 0 0 0 0 0 5000 0 10000 0 0
SOLVE,K,R
PRINT,R
MEMFRC,T1,R,LM,F1,1
PRINT,F1
MEMFRC,T2,R,LM,F2,2
PRINT,F2
MEMFRC,T3,R,LM,F3,3
PRINT,F3
MEMFRC,T4,R,LM,F4,4
PRINT,F4
MEMFRC,T5,R,LM,F5,5
PRINT,F5
MEMFRC,T6,R,LM,F6,6
PRINT,F6
MEMFRC,K1,R,LM,G1,1
PRINT,G1
MEMFRC,K2,R,LM,G2,2
PRINT,G2
MEMFRC,K3,R,LM,G3,3
PRINT,G3
MEMFRC,K4,R,LM,G4,4
PRINT,G4
MEMFRC,K5,R,LM,G5,5
PRINT,G5
MEMFRC,K6,R,LM,G6,6
PRINT,G6
NODES,JOINTS,6,1
1 0 0 0 1 1
1 0 150 0 3
4 300 300 0 1 0
5 350 150 0 1 0
```

```
6 400 0 0 1 0
0 0 0 0 0 0 0
LOADI.CONN.6.3.9
2 1 2
2 2 3
2 3 4
2 2 5
2 4 5
2 5 6
GRAPH
YES
TITLE.2
PLST.JOINTS.CONN.1
READ.5
```

b. Two Dimensional Truss Analysis

This example was published as an example problem in [Ref. 6: pp. 150-153]. The geometry of the problem is shown in Figure 10. As expected, CALNPS provided the exact same results as the source. Due to its length the output data file is not presented here, however, this file is also available on the online demonstration facility.

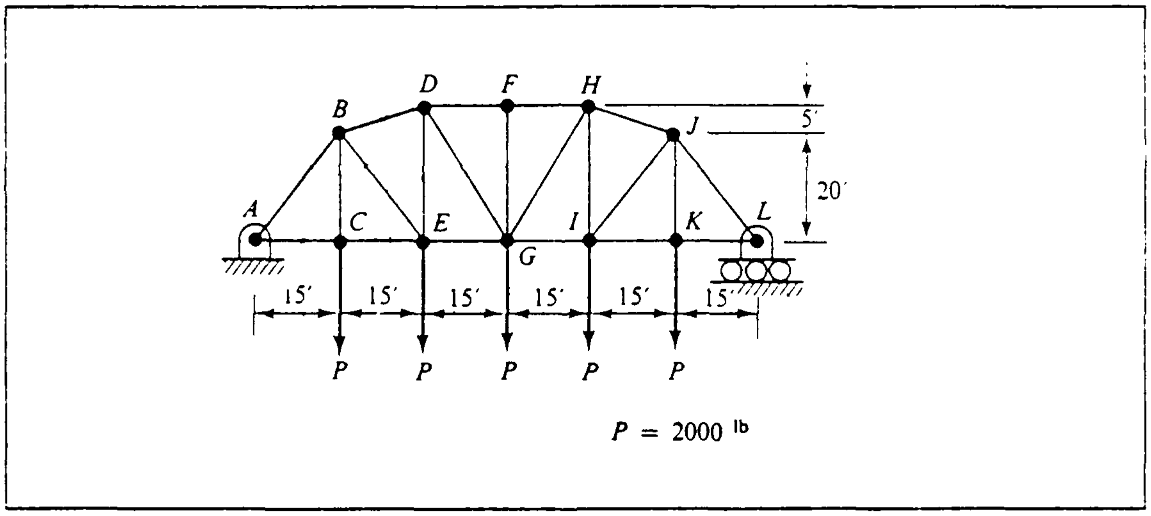


Figure 10. Sample Truss Problem.

(1) Input Data File.

```
START
NODES.JOINTS,12.1
1 0 0 0 1 1
1 15 0 0 7
8 15 20 0 1 0
9 30 25 0 1 1
1 15 0 0 11
```

```
12 75 20 0 1 0
0 0 0 0 0 0 0 0
BOUND,BC
2 6 1 1 0 0 0 0 1
7 7 1 0 0 0 0 0 0
8 12 1 1 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0
LOAD,PROPERTIES,1,3,9
3000 3000000 .675
TRUSS,TR1,JOINTS,BC,PROPERTIES
1 1 8 1
2 1 2 1
3 2 8 1
4 8 9 1
5 8 3 1
6 2 3 1
7 3 9 1
8 9 10 1
9 9 4 1
10 3 4 1
11 10 4 1
12 10 11 1
13 4 11 1
14 4 5 1
15 5 11 1
16 11 12 1
17 5 12 1
18 5 6 1
19 6 12 1
20 12 7 1
21 6 7 1
0 0 0 0
ADDSF,RIGID,MAS
LOADS,LDS,BC,1
2 1 0 -2000 0 0 0 0
3 1 0 -2000 0 0 0 0
4 1 0 -2000 0 0 0 0
5 1 0 -2000 0 0 0 0
6 1 0 -20000 0 0 0
0 0 0 0 0 0 0 0 0
SOLVE,RIGID,LDS
DISPL,LDS,BC
FORCE,TR1,LDS,M3
LOADI,CONN,21,3,9
2 1 8
2 1 2
2 2 8
2 8 9
2 8 3
2 2 3
2 3 9
2 9 10
```

```
2 9 4
2 3 4
2 10 4
2 10 11
2 4 11
2 4 5
2 5 11
2 11 12
2 5 12
2 5 6
2 6 12
2 12 7
2 6 7
GRAPH
YES
TITLE,2
PLST,JOINTS,CONN,1
READ,5
```

c. Static Analysis of a Bridge

This example demonstrates the static analysis of a three dimensional bridge. The problem is to solve for the static displacement and member forces due to a 1K force at joint number four in the X direction. The geometry of the structure and the properties are shown below in Figure 11.

(1) Input Data File.

```
START
LABEL,3
                    **********************************
                    ** STATIC ANALYSIS EXAMPLE **
                    **********************************
NODES,JOI,4          ** GEOMETRY OF THE STRUCT. **
1,0,0,400
2,0,1600,0
3,0,2800,400
4,0,1600,400
0,0,0,0
PRINT,JOI,,4      ** NODE NUMBERS AND COORDINATES **
BOUND,BC           ** NON ZERO DISPLACEMENTS **
4,4,1,1,1,1,1,1,1
0,0,0,0,0,0,0,0,0
LOAD,BPR,2,7,9    ** BEAM PROPERTIES **
3000,1100000,2000000,700000,3000000,1200000,0.675
1000,116000,82500,82500,3000000,1200000,.225
PRINT,BPR,,4
BEAM,BRIDGE,JOI,BC,BPR
1,1,4,2,1
2,4,3,2,1
3,4,2,3,2
0,0,0,0,0
```

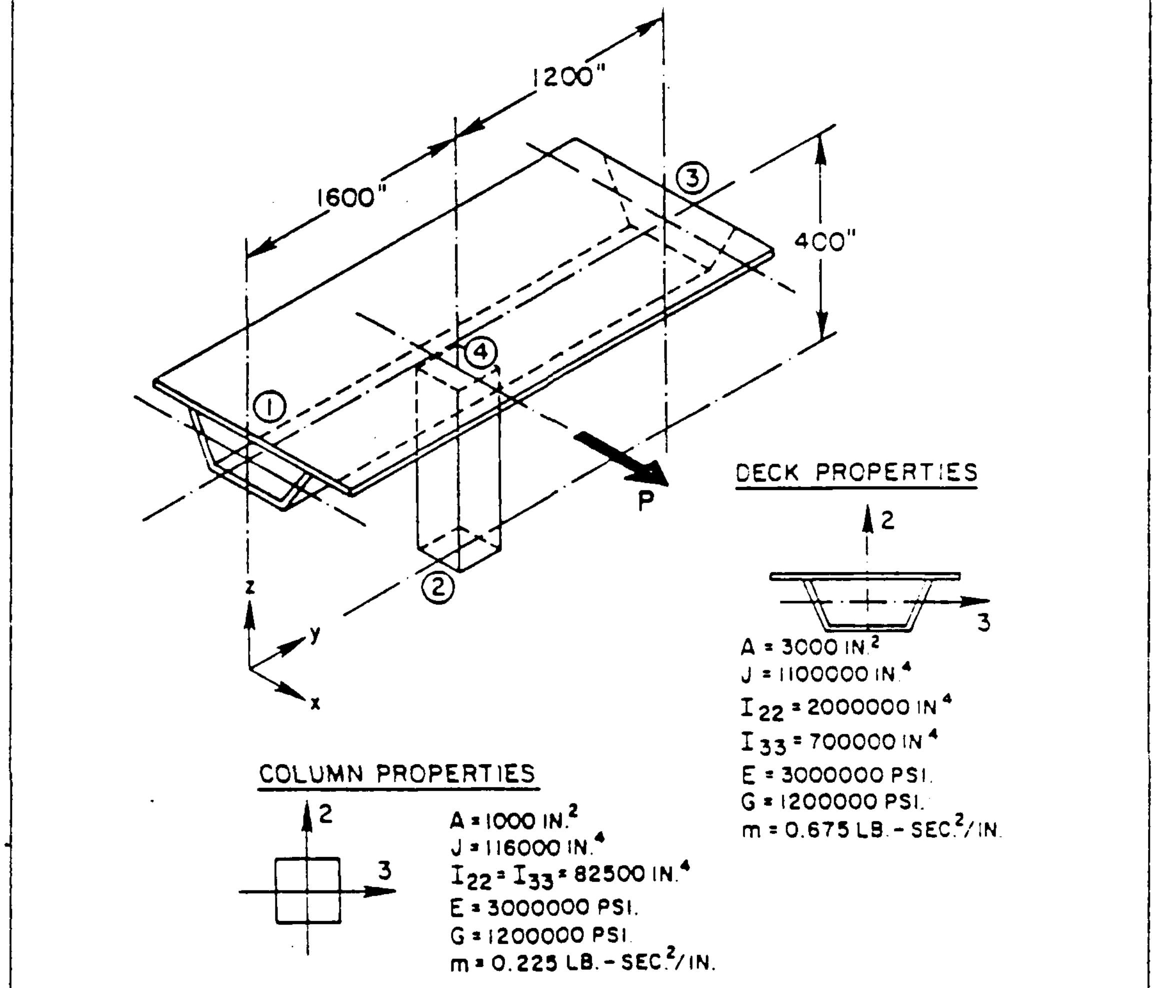


Figure 11.   Static Analysis of a Bridge.

```
ADDSF,K,MAS            ** FORM STIFFNESS AND MASS MATRICES **
PRINT,K,,4        ** STIFFNESS MATRIX **
LOADS,LDS,BC,1    ** LOAD VECTOR **
4,1,1000,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0
SOLVE,K,LDS
DUP,LDS,LLDS
DISPL,LLDS,BC          ** DISPLACEMENTS **
FORCE,BRIDGE,LDS  ** MEMBER FORCES **
READ,5
```

(2) *Solution.* The output file for this problem is available through the online demonstration facility. The solution of the forces is as shown in Figure 12 below.

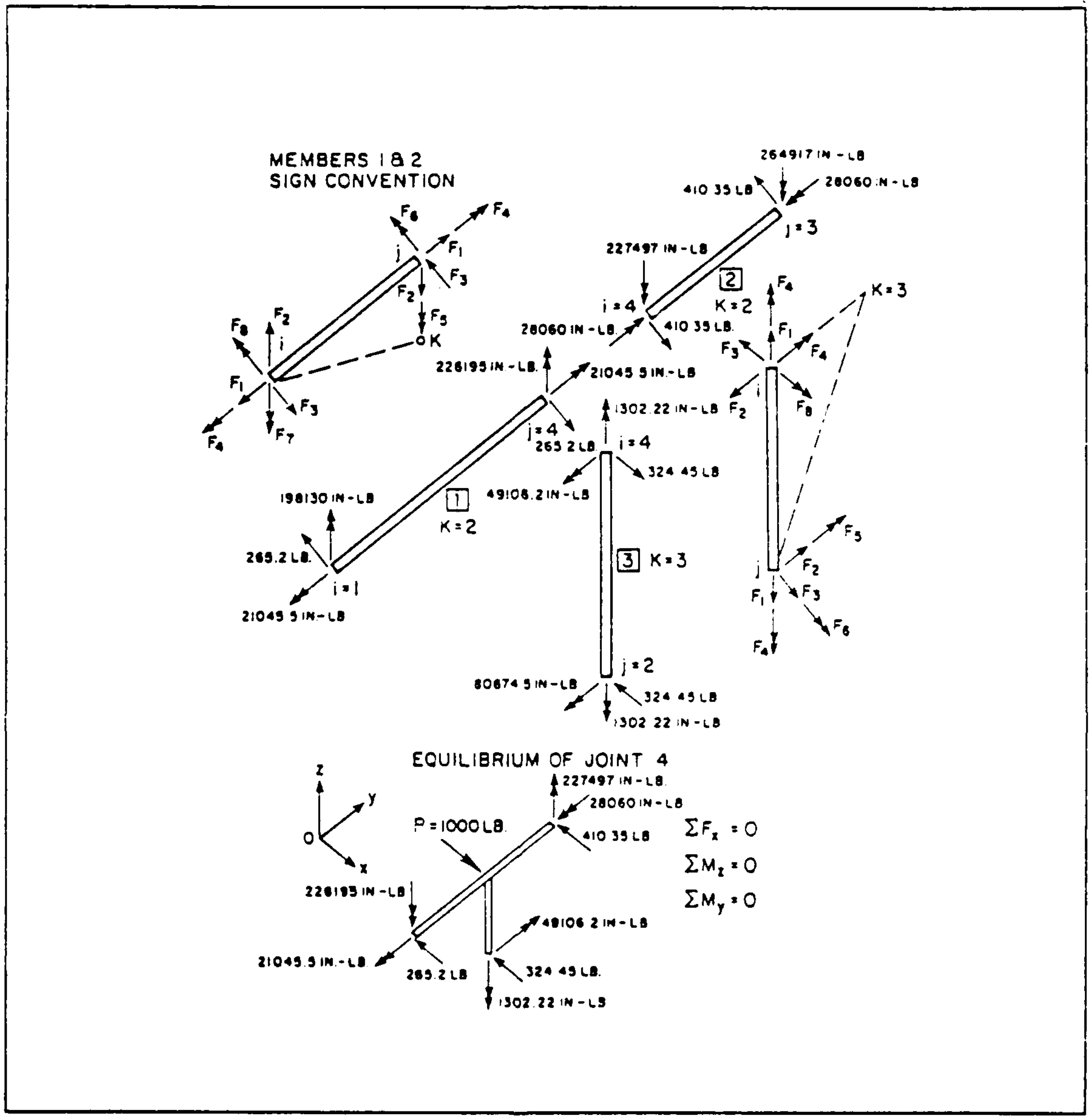


Figure 12. Solution of Bridge Problem (Static).

D. DYNAMIC ANALYSIS OPERATIONS

The dynamic analysis operations were not changed within the scope of this work.

1. Description

The dynamic analysis operations were designed to evaluate the dynamic response of structures subjected to arbitrary time-dependent loads. These operations work in close conjunction with the static analysis operations. The user has the option of using the mode superposition method or a direct step by step integration of the dynamic equations of motion. The user may examine the spectra of both input loading and calculated displacements.

The most common and convenient form for time-dependent data to be specified is as straight line segments between given time points. Therefore, an operation which generates values at equal intervals is necessary. Another common characteristic of time-varying loads on structures is that it is normally possible to represent the loads at all points on the structure by the product of two matrices, a column matrix indicating the spatial distribution of loads times a row matrix which indicates the values as a function of various times. If a more complicated loading is required, it is possible to perform more analyses, each within the restrictions of the program, then add the results of each analysis.

2. Command Specifications

<u>DYNAM,M1,M2,M3,M4,$\overset{+}{\text{M5}}$,M6,N1</u>

This operation evaluates the following set of uncoupled second order differential equations associated with the mode superposition method for the dynamic analysis of a structural system:

For i = 1 to N nodes:

$$\ddot{x}_i + 2\lambda_i\omega_i\dot{x}_i + \omega_i^2 x_i = P_i(t)$$

M1 M1 is the name of a row or column matrix which contains N terms (frequencies in rad sec).

M2 M2 is the name of a row or column matrix containing the N λ(i) terms (ratio of modal damping to critical damping).

The generalized time-varying forces P_i(t) are not specified directly, but are evaluated from more fundamental information. The forces for all modes are evaluated at specific times by the program from the following matrix equation:

$$[P] = [P][F] = [M3][M4]$$

M3 P is a specified N X 1 vector named M3.

M4 F is a 1 X N1 row matrix generated from the 2 X K array named M4. M4 will have the following format:

$$[M4] = \begin{bmatrix} t_1 & t_2 & \dots & t_K \\ F_1 & F_2 & \dots & F_K \end{bmatrix}$$

	M4 is the same form as the input array described under the operation, FUNG. It is not necessary to use the FUNG operation before the DYNAM operation.
M5	M5 is the name of the N X N1 array which contains the generalized displacement $x_i(t)$.
M6	M6 is the name of the 1 X 1 array which contains the time increment associated with the displacements.
N1	N1 is the number of displacements to be generated. The method of integration is exact for straight line segments.

EIGEN,$\overset{-}{\mathrm{M}}$1,$\overset{+}{\mathrm{M}}$2,$\overset{-}{\mathrm{M}}$3, ,N1

This operation solves the following eigenvalue problem:

$$K\phi = M\phi\lambda$$

M1	M1 is the N X N, symmetric, positive definite matrix, K.
M2	M2 contains the eigenvectors, ϕ, for the eigenvalues in M3. Eigenvectors are stored in the columns corresponding to respective eigenvalues.
M3	M is a diagonal matrix of nonzero, positive terms designated by M3. M3 must be a row or column matrix containing only the diagonal terms of M.
N1	N1 specifies the approximate number of significant figures of the eigenvalues. The maximum accuracy possible is 16 and the default value is 4. The use of more than 12 figure accuracy is not recommended.

The program uses the standard Jacobi diagonalization method to solve for all eigenvalues and eigenvectors.

FUNG,M1,$\overset{+}{\mathrm{M}}$2,M3,N1,N2

This operation generates a matrix named M2 which contains values at equal intervals of the function specified in the array named M1. M1 must be a 2 X K array of the form:

$$[M1] = \begin{bmatrix} t_1 & t_2 & \cdots & t_K \\ F_1 & F_2 & \cdots & F_K \end{bmatrix}$$

which numerically represents a function of the form shown below in Figure 13.

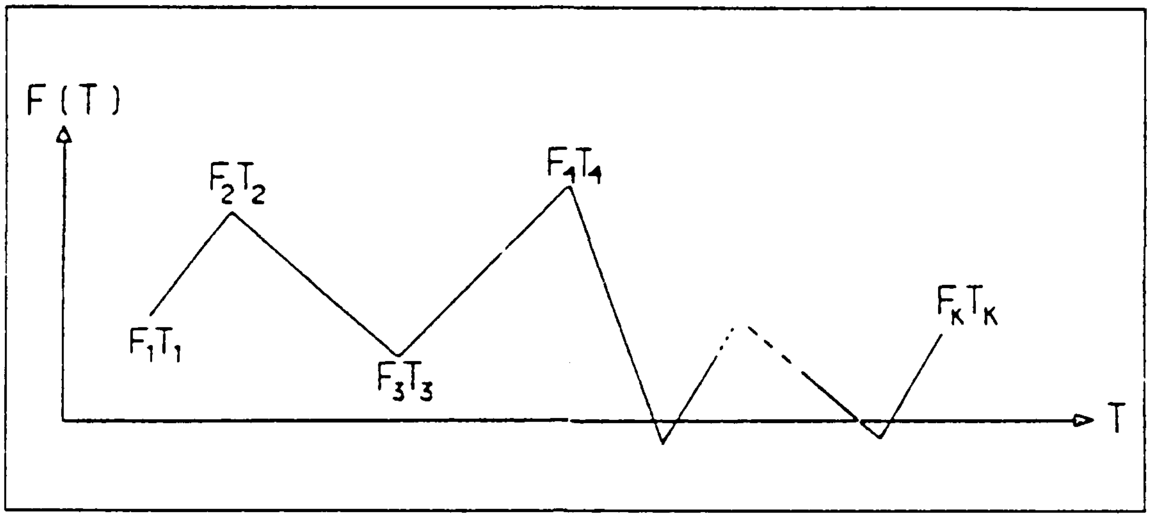


Figure 13. Form of the Function Represented by M1.

K = N1 N2 where N1 and N2 are defined by the STEP operation.

M3	The time interval is specified in the 1 X 1 matrix named M3.
N1	N1 specifies the total number of values to be generated and is the number of columns in M2.
N2 = 0	The array will be a 1 X N1 row matrix in which the first value will be F.
N2	If N2 is not equal to zero, the array M2 will be a 2 X N1 matrix of the following form:

$$[M2] = \begin{bmatrix} t_1 & t_1 + \Delta t & t_1 + 2\Delta t & \dots \\ F_1 & F(t_1 + \Delta t) & F(t + 2\Delta t) & \dots \end{bmatrix}$$

PLOT.M1.N1

This operation will prepare a printer plot of selective rows of the matrix named M1. N1 is the number of rows of M1 which will be plotted by this operation. The PLOT command must be followed by N1 data lines in (1A1,I4) format with the following information:

ITEM	CONTENTS
1	The plot symbol which can be any keypunch symbol. The symbol must be enclosed within quotes.
2	The row number to be plotted.

Example: PLOT.M1.1 followed by: '*',1

CALNPS automatically searches the information to be plotted for maximum and minimum values. The difference in these numbers divided by 120 spaces is selected as the plot scale.

STEP,$\overset{-}{\text{M}}$1,M2,M3,$\overset{-}{\text{M}}$4, $\overset{+}{\text{M}}$5,M6,M7,M8,N1,N2

This operation calculates the dynamic response of a structural system using direct step by step of the following linear matrix equation of motion:

$$[M][A] + [C][V] + [K][U] = [R(t)] = [P][F(t)]$$

where:

A	Acceleration vector.
V	Velocity vector.
M1	M1 is the name of the N X N stiffness matrix, K.
M2	M2 is the name of the N X N mass matrix, M.
M3	M3 is the name of the N X N damping matrix, C.
M4	M4 is the name of the N X 3 initial condition matrix, U, in which: U(I,1) is a vector of displacements, U. U(I,2) is a vector of velocities, V. U(I,3) is a vector of accelerations, A.
M5	M5 is the name of the N X N2 matrix of calculated displacements in which column I represents the displacements at time I*N1* Δ t.
M6	M6 is the name of the N X 1 load distribution matrix, P.
M7	M7 is the name of the 1 X K row matrix representing the load multipliers at equal time increments F, where K = N2 N1.
M8	M8 is the name of the 1 X 1 matrix containing Δ t.
N1	N1 is the output interval for the displacements.
N2	N2 is the total number of displacement vectors to be calculated.

The total time for which results will be calculated by this operation is $N1 \times N2 \times \Delta t$. This command must be followed by data lines containing the following information in free format:

ITEM	CONTENTS
1	δ
2	α
3	θ

Different values of δ, α and θ will allow the user to select different methods of step by step integration. The following are some possibilities:

	δ	α	θ
Newmarks average acceleration	1/2	1/4	1.0
Linear acceleration	1/2	1/6	1.0
Wilson's θ method (low damping)	1/2	1/6	1.42
Wilson's θ method (hi damping)	1/2	1/6	2.0

3. Examples

a. *Dynamic Analysis of a Bridge*

This example is a continuation of the bridge problem in the previous section, where a static analysis was done. Refer to Figure 11 on page 49 to review the geometry of the problem. This example demonstrates how the dynamic commands work in conjunction with the static commands. Note that the commands in the input file are identical up to a point.

(1) *Input Data File.*

```
START
LABEL,1
               ** DYNAMIC ANALYSIS EXAMPLE **
NODES,JOI,4    ** GEOMETRY OF THE STRUCTURE **
1,0,0,400
2,0,1600,0
3,0,2800,400
4,0,1600,400
0,0,0,0,0,0,0
PRINT,JOI                   ** NODE NUMBERS AND COORD. **
BOUND,BC                    **   NON ZERO DISPLACEMENTS  **
4,4,1,1,1,1,1,1,1
0,0,0,0,0,0,0,0,0,0,0
LOAD,BPR,2,7,9                  ** BEAM PROPERTIES **
3000,1160000,2000000,700000,3000000,1200000,0.675
1000,116000,82500,82500,3000000,1200000,0.225
PRINT,BPR,,4
BEAM,BRIDGE,JOI,BC,BPR
1,1,4,2,1
2,4,3,2,1
3,4,2,3,2
0,0,0,0,0,0
ADDSF,K,MAS            ** FORM STIFFNESS AND MASS MATRICES **
DUP,K,KK
ZERO,MASS,6,6,0,0
STODG,MASS,MAS
PRINT,MASS,,3
ZERO,C,6,6,0,0
LOAD,DELTA,1,1,9
0.05
```

```
PRINT,DELTA                       ** INCREMENT OF TIME **
LOAD,FUA,2,4,9
0,1,2,5
0,20000,0,0
PRINT,FUA                      ** TIME DEPENDENT LOAD **
FUNG,FUA,FUAI,DELTA,500,0
LOAD,SCALEP,6,1,9
1,0,0,0,0,0
PRINT,SCALEP
ZERO,INCO,6,3,0,0
STEP,K,MASS,C,INCO,DISP,SCALEP,FUAI,DELTA,1,100
.5,.1666666666666667,1.42
PLOT,DISP,1                   ** X DISPLACEMENT AT JOINT 4 **
'*',1
EIGEN,KK,MODES,MAS
PRINT,MODES,,4                     ** MODE SHAPES **
SQREL,MAS
PRINT,MAS                         ** FREQUENCIES **
LIST
READ,5
```

(2) *Output.* The complete solution is available through the online demonstration facility. Figure 14 illustrates the display of the displacements produced by the PLOT command.

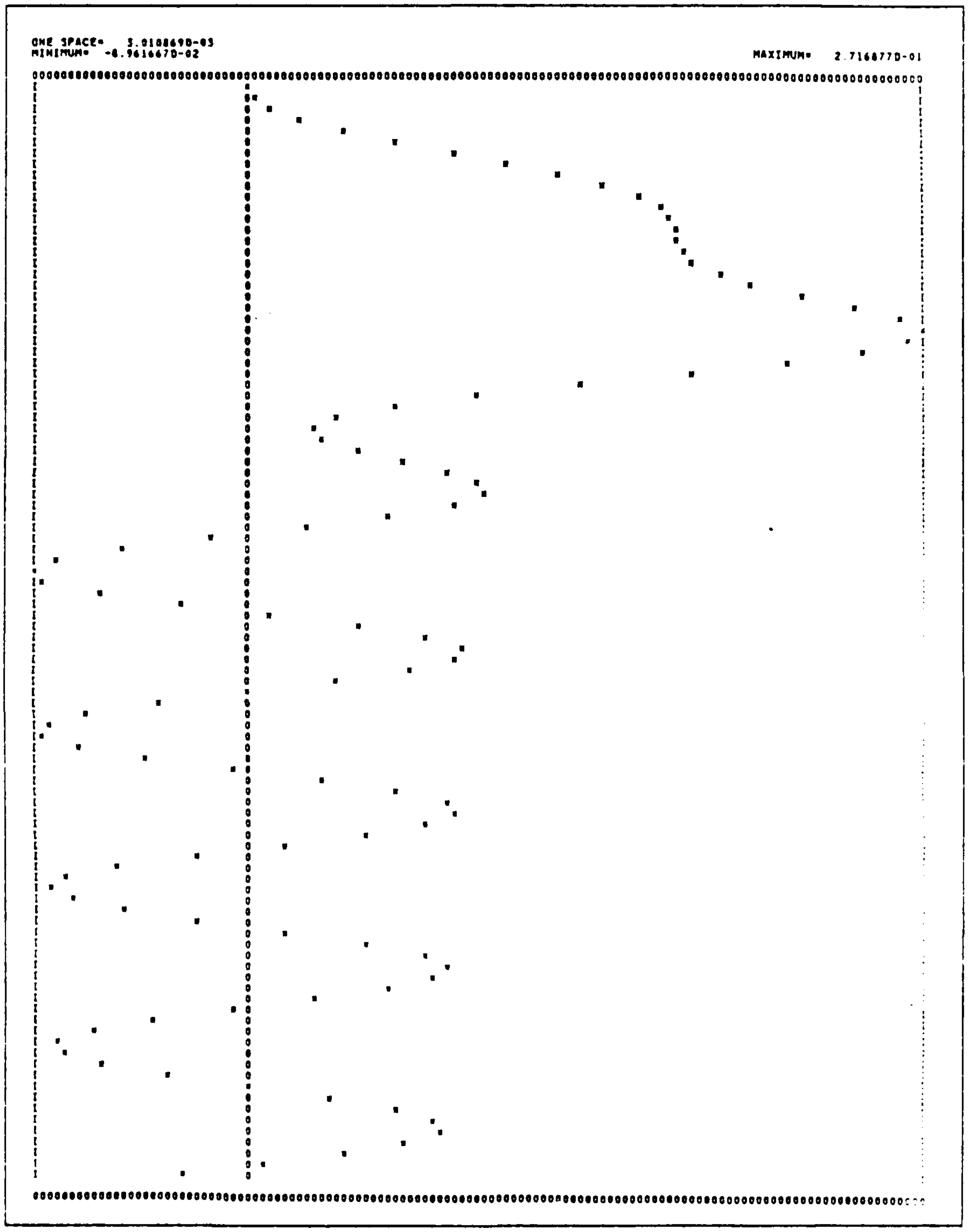


Figure 14. Displacements of Bridge Problem.

E. HEAT TRANSFER OPERATIONS

This section is a revision of the material published by Roberts in [Ref. 3: pp. 63-72]. This series of operations was added as a result of the integration of FEAP with CALNPS.

1. Description

The heat transfer operations serve to form the total conductivity and heat capacity matrices for systems of two or three dimensional elements, to form the flux vector and solve the defined set of equations. For two dimensional elements, isoparametric elements of four to nine nodes are available. For three dimensional elements, isoparametric elements of eight to 21 nodes are available.

A heat transfer problem must first be initialized with the command, HTXFR. The geometry of the problem is then set up with the commands, COORD and ELCON. COORD creates an array containing the coordinates of the system nodes and ELCON specifies the element connectivity. Material properties are next specified by the command, PROP. Nodes of constant temperature are specified with the command, CTEMP. The equation profile for the problem is generated by the command, PROF.

After the problem geometry, properties, boundary conditions and equation profile are established as above, the conductivity matrix can be formed with the command, SYMC or USYMC. The heat capacity matrix can be approximated with either a constant or lumped matrix formulation with the commands, CCAP or LCAP respectively. The flux vector is formed with the command, FORM.

The equations are solved with the appropriate equation solver. Time independent systems are solved with the command, CALC. Time dependent systems are solved with the command, ODE. Finally, the nodal temperatures can be printed with the command, PTEMP.

2. Command Specifications

ADTIM

This operation advances the time in a heat transfer problem by the amount input with the command, DTIM.

CALC

This operation solves time independent heat transfer problems for temperature and updates the temperature matrix.

CCAP

This operation forms a consistent capacitance matrix for heat transfer systems.

CONV

This operation performs a temperature convergence test on a heat transfer system. If this is used inside a loop (LOOP operation) and the test shows convergence, looping will be terminated.

COORD

This operation creates an array (NDM,NUMNDP) which contains the coordinates for all nodes in a heat transfer system. The variables NDM and NUMNDP are defined by the HTXFR command, where NDM represents the spatial dimension and NUMNDP represents the number of nodes in the mesh. Positive coordinate systems are as shown in Figure 15. Data lines containing the following information must be entered in free format immediately following the COORD command:

ITEM	CONTENTS
1	Node number.
2	X-coordinate.
3	Y-coordinate.
4	Z-coordinate.
5	System type (See note 1 below).
6	Generation code. (See note 2 below).

Note 1: System type refers to the coordinate system used when inputting the data. All coordinates will be converted to the Cartesian coordinate system for use by CALNPS. The system type codes are as listed below:

SYSTEM TYPE	SYSTEM
0,1	Cartesian
2	Cylindrical, Z axis longitudinal.
3	Cylindrical, Y axis longitudinal.
4	Cylindrical, Z axis longitudinal.
5	Spherical.

Angle data should be entered in degrees. The input data is the same as above with the following correspondence:

Cartesian	Cylindrical	Spherical
X	ρ	ρ
Y	θ	θ
Z	Z	ϕ

Note 2: If the generation code is not zero, the next data line is a generation vector for the self generation of nodes. It is formatted as follows:

ITEM	CONTENTS
1	Node number increment.
2	X increment.
3	Y increment.
4	Z increment.
5	Last node number to be generated.

It is assumed that the increments pertain to the same system of reference as the previous line of data. This operation must be terminated by a row of at least six zeroes separated by spaces or commas.

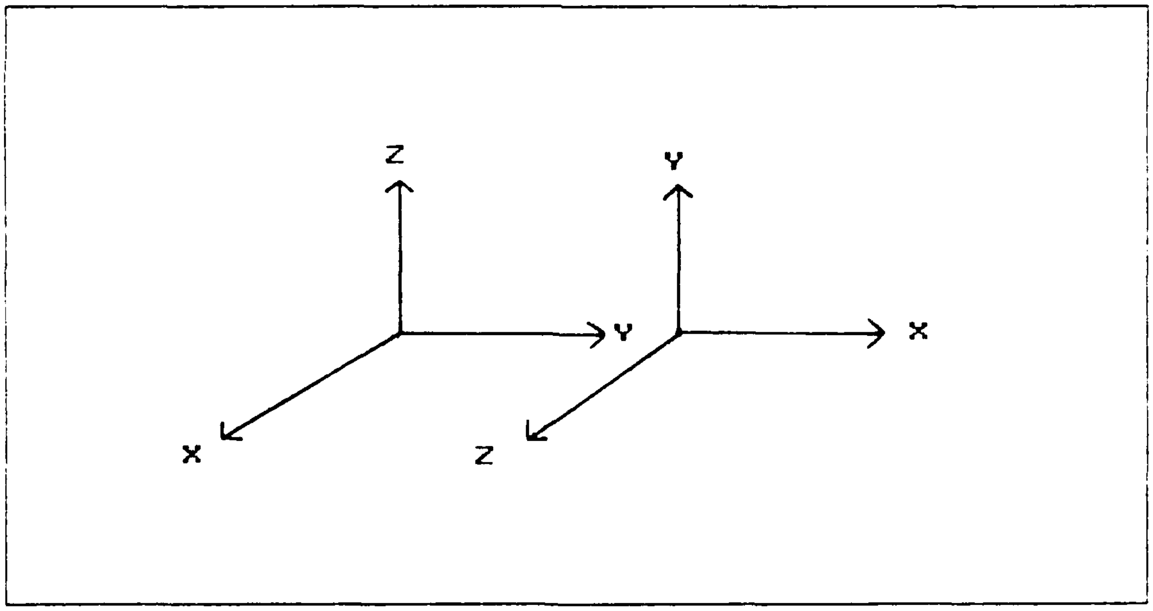


Figure 15. Positive Coordinate Systems for COORD Command.

CTEMP

This operation inputs constant temperature boundary restraint data. Temperatures for nodes with constant temperatures must be entered. For nodes with no temperature restriction, no entries should be made. Generation capability is built in with data entered in free format as follows:

ITEM	CONTENTS
1	Initial node.
2	Last node.
3	Node increment.
4	Temperature.

This operation must be terminated by a row of at least four zeros separated by spaces or commas.

DTIM,M1

This operation sets the time increment for integration in a heat transfer system to the value found in the 1 X 1 matrix named M1.

EIGV

This operation computes the dominant eigenvalue and vector of the current heat transfer conductance matrix.

ELCON

This operation creates an array (NEN + 1,NUML) which contains the element connectivity data for all elements in a heat transfer system where:

NEN + 1 = number of nodes per element.
NUML = number of elements.

Data lines containing the following information must be entered in free format immediately following the ELCON command:

ITEM	CONTENTS
1	Element number.
2	Node 1 number.
3	Node 2 number.
ETC.	ETC.

N + 1	Node N number.
N + 2	Material set number.
N + 3	Generation code.

If the generation code is not zero, the next data line is a generation vector for the self generation of element connectivity. It is entered in free format as follows:

ITEM	CONTENTS
1	Element number increment.
2	Node 1 increment.
3	Node 2 increment.
ETC.	ETC.
N + 1	Node N increment.
N + 2	Material set increment (usually zero).
N + 3	Last element number to be generated.

The conventions for connectivity are as shown in Figure 16 below. This operation must be terminated by a row of at least NEN + 1 zeroes separated by spaces or commas.

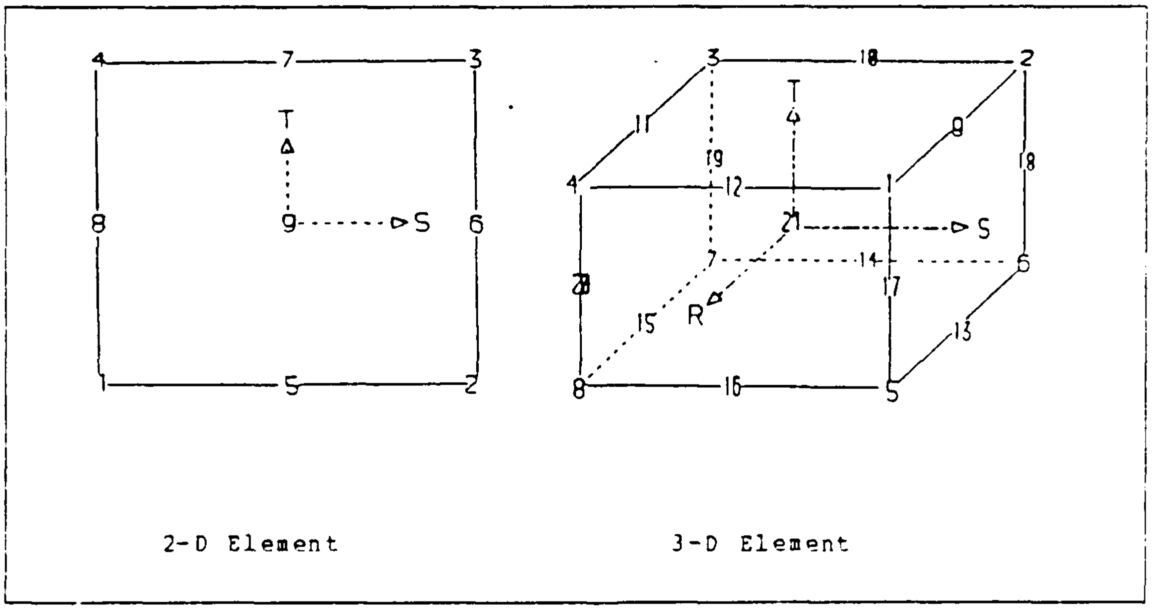


Figure 16. Connectivity Conventions for Heat Transfer Elements.

FORM

This operation forms the flux vector for heat transfer systems of equations.

HTXFR

This operation initializes the heat transfer problem solving group. The following information entered in free format must immediately follow this command:

ITEM	CONTENTS
1	Number of nodes (NUMNDP).
2	Number of elements (NUML).
3	Number of material sets (NUMMAT).
4	Spatial dimension (2 or 3) (NDM).
5	Number of unknowns per node (NDF).
6	Maximum number of nodes per element (NEN).

LCAP

This operation forms a lumped capacitance matrix for heat transfer systems.

ODE,M1

This operation is the first order differential equation solver for heat transfer systems. Initial conditions are established by issuing the command, ODE,INIT. The ODE command is then issued again to define the type of integration scheme to be used. If M1 is "LINE", then a two point integration scheme is used and if M1 is "QUAD", a three point integration scheme is used. Data lines containing the following information must be entered in free format immediately following the ODE,INIT command:

ITEM	CONTENTS
1	Integration parameter, θ, for two point scheme. The default is 2/3.
2	Integration parameter, γ, for three point scheme. The default is 1.5.
3	Integration parameter, β, for three point scheme. The default is 0.8.
4	Maximum temperature allowed.
5	Minimum temperature allowed.

Default values are obtained by entering zeros for items 1, 2 and 3 above.

To establish the initial temperatures, the following data should be entered immediately following the above data in free format:

ITEM	CONTENTS
1	Node number.
2	Initial temperature.
3	Generation code.

If the generation code is not equal to zero, then the next data line is a generation vector for the self generation of initial nodal temperatures. The data should contain the following information entered in free format:

ITEM	CONTENTS
1	Node number increment.
2	Temperature increment.
3	Last node to be initialized.

Repeatedly enter the above data until all nodes have been initialized. If a node is not initialized, it will have an initial temperature equal to zero. This operation must be terminated by a row of at least three zeros separated by commas or spaces.

PRLD

This operation allows a simplified proportional loading for heat transfer systems. *The loading is calculated by:*

$$PROP = A1 + A2 \times TIME + A3(\sin(A4 \times TIME + A5))^L$$

The coefficients, exponents and time limits should be entered in free format as follows:

ITEM	CONTENTS
1	L
2	Minimum time for which prop is calculated.
3	Maximum time for which PROP is calculated.
4	A1
5	A2
6	A3
7	A4
8	A5

PROF

This operation establishes the profile of the equations for solution of the problem. After the issuance of this command, the problem is set and the node numbers with restrained boundary conditions (constant temperatures) may not be changed. The value of the restrained temperatures may be changed.

PROMPT

This operation permits the prompts for user supplied data to be suppressed without the loss of other output. A second issuance of the command, PROMPT will restore the prompts unless the output is suppressed with the NO command.

PROP

This operation inputs the material property data for a heat transfer system. The following information, entered in free format, must follow this command:

ITEM	CONTENTS
1	Material set number.
2	Element type number (2 = 2-D; 3 = 3-D).

The material property data must be entered in consistent units as shown below:

Parameter	English Units	SI Units
k	$\frac{BTU}{(HR \times FT \times {}^{\circ}F)}$	$\frac{W}{(M \times C)}$
k	$\frac{BTU}{(HR \times FT \times {}^{\circ}F)}$	$\frac{W}{(M \times C)}$
C	$\frac{BTU}{(lbm \times {}^{\circ}F)}$	$\frac{KJ}{(KG \times C)}$
ρ	$\frac{LBM}{FT^3}$	$\frac{KG}{M^3}$
Q'''	$\frac{BTU}{(HR \times FT^3)}$	$\frac{W}{M^3}$

Additional information must be provided depending on the spatial dimension of the element being used. This information should be entered as follows immediately following the PROP command:

For two dimensional elements, the following information must follow the PROP command. An entry must be made for each item. If the item is temperature dependent, the entry will be ignored.

ITEM	CONTENTS
1	Conductivity in the X-direction.
2	Conductivity in the Y-direction.
3	Specific heat.
4	Density.
5	Heat generation per unit volume.
6	Number of integration points per direction (one to six). The default is four.
7	Geometry type (1 = plane geometry; 2 = axisymmetry).
8	Total number of lines with specified boundary conditions in elements with the same material set number (NLBC). See Note 1 below.
9	Temperature dependence code. See Note 2 below.

Note 1: If any lines have a specified boundary condition (NLBC > 0), a data line must be entered for each line. If the same line is subjected to more than one type of boundary condition, a data line must entered for each one of these types. The total number of additional lines must equal NLBC. The information should be entered in free format as follows:

ITEM	CONTENTS
1	Element number.
2	Boundary condition code (see Note 3 below).
3	Line code (see Note 4 below).
4	Property value (see Note 5 below).
5	Ambient temperature (see Note 6 below).

Note 2: The temperature dependence code is zero if all material properties are constant and one if any property is temperature dependent. If the code is one, the following information is required:

ITEM	CONTENTS
1	Conductivity in X-direction code.
2	Conductivity in Y-direction code.
3	Heat capacity (specific heat X density) code.
4	Heat generation per unit volume code.

Zero represents a constant property and one represents a temperature dependent property. Temperature dependent properties are entered in the form of a table. The tables are consecutively input for conductivity in the X-direction, conductivity in the Y-direction, heat capacity and heat generation per unit volume. Omit the tables for which the temperature dependence code is zero. Tables are input in free format as follows:

ITEM	CONTENTS
1	Number of data pairs to be entered.
2	Temperature 1.
3	Property 1.
4	Temperature 2.
5	Property 2.
ETC.	ETC.
2N	Temperature N.
2N + 1	Property N.

Note 3: Boundary condition codes are as follows:

1	Flux.
2	Convection (constant coefficient).
3	Radiation.
4	Convection (temperature dependent property). See Note 7 below.

Note 4: The line subjected to a specified boundary condition is identified using the local coordinates. A line is numbered one or two if it is perpendicular to the axis S or T respectively. These numbers are positive or negative according to which direction of the axis it is perpendicular. Figure 17 shows the orientation of the S and T axes. The line codes are as follows:

1	S = +1 line.
-1	S = -1 line.
2	T = +1 line.
-2	T = -1 line.

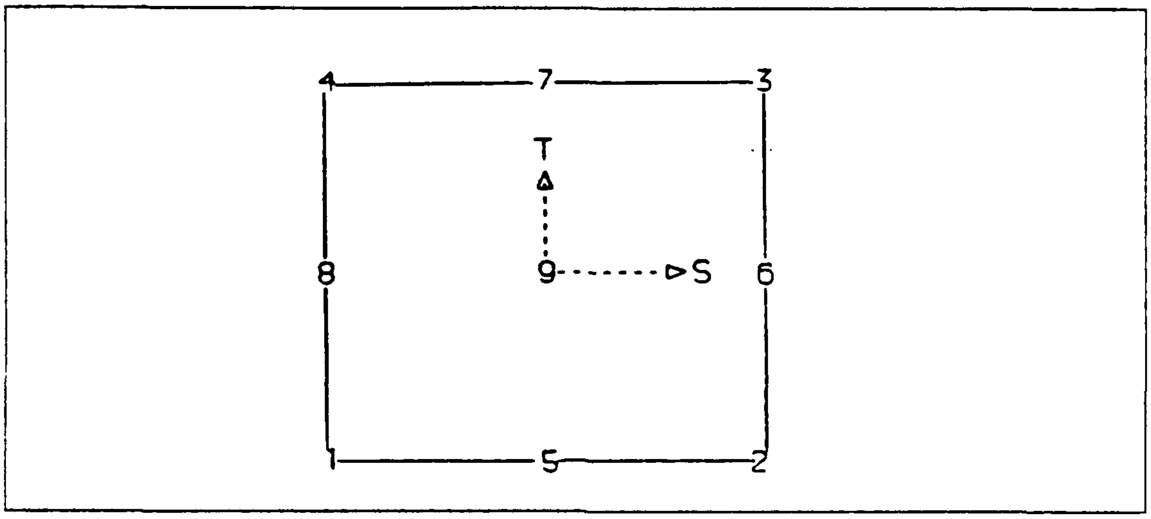


Figure 17. S and T Axes for Two Dimensional Heat Transfer Elements.

Note 5: The property values are as follows:

Flux-Flux per unit area.
Convection-Constant heat transfer coefficient (ignored if temperature dependent).
Radiation - Product of emissivity by Stephan-Boltzman constant.

Note 6: The ambient temperature is ignored *for the flux boundary condition.*

Note 7: If the boundary condition code is four (temperature dependent heat transfer coefficient), a table for the temperature dependence must be entered in free format and contain the following information:

ITEM	CONTENTS
1	Number of data pairs to be entered.
2	Temperature 1.
3	Heat transfer coefficient 1.
4	Temperature 2.
5	Heat transfer coefficient 2.
ETC.	ETC.
2N	Temperature N.
2N + 1	Heat transfer coefficient N.

For three dimensional elements, the following information must follow the PROP command. An entry must be made for each item. If the item is temperature dependent, the entry will be ignored.

ITEM	CONTENTS
1	Conductivity in the X-direction.
2	Conductivity in the Y-direction.
3	Conductivity in the Z-direction.
4	Specific heat.
5	Density.
6	Heat generation per unit volume.
7	Number of integration points per direction (one to six). The default is four.
8	Geometry type (1 = plane geometry; 2 = axisymmetry).
9	Total number of surfaces with specified boundary conditions in elements with the same material set number (NLBC). See Note 1 below.
10	Temperature dependence code. See Note 2 below.

Note 1: If any surfaces have a specified boundary condition (NSBC > 0), a data line must be entered for each line. If the same line is subjected to more than one type of boundary condition, a data line must entered for each one of these types. The total number of additional lines must equal NSBC. The information should be entered in free format as follows:

ITEM	CONTENTS
1	Element number.
2	Boundary condition code (see Note 3 below).
3	Surface code (see Note 4 below).
4	Property value (see Note 5 below).
5	Ambient temperature (see Note 6 below).

Note 2: The temperature dependence code is zero if all material properties are constant and one if any property is temperature dependent. If the code is one, the following information is required:

ITEM	CONTENTS
1	Conductivity in X-direction code.
2	Conductivity in Y-direction code.

3	Conductivity in Z-direction code.
4	Heat capacity (specific heat X density) code.
5	Heat generation per unit volume code.

Zero represents a constant property and one represents a temperature dependent property. Temperature dependent properties are entered in the form of a table. The tables are consecutively input for conductivity in the X-direction, conductivity in the Y-direction, conductivity in the Z-direction, heat capacity and heat generation per unit volume. Omit the tables for which the temperature dependence code is zero. Tables are input in free format as follows:

ITEM	CONTENTS
1	Number of data pairs to be entered.
2	Temperature 1.
3	Property 1.
4	Temperature 2.
5	Property 2.
ETC.	ETC.
2N	Temperature N.
2N + 1	Property N.

Note 3: Boundary condition codes are as follows:

1	Flux.
2	Convection (constant coefficient).
3	Radiation.
4	Convection (temperature dependent property). See Note 7 below.

Note 4: The surface subjected to a specified boundary condition is identified using the local coordinates. A surface is numbered one, two or three if it is perpendicular to the R, S or T axis respectively. These numbers are positive or negative according to which direction of the axis it is perpendicular. Figure 18 shows the orientation of the R, S and T axes. The line codes are as follows:

1	R = +1 surface.
-1	R = -1 surface.
2	S = +1 surface.
-2	S = -1 surface.

3	T = +1 surface.
-3	T = -1 surface.

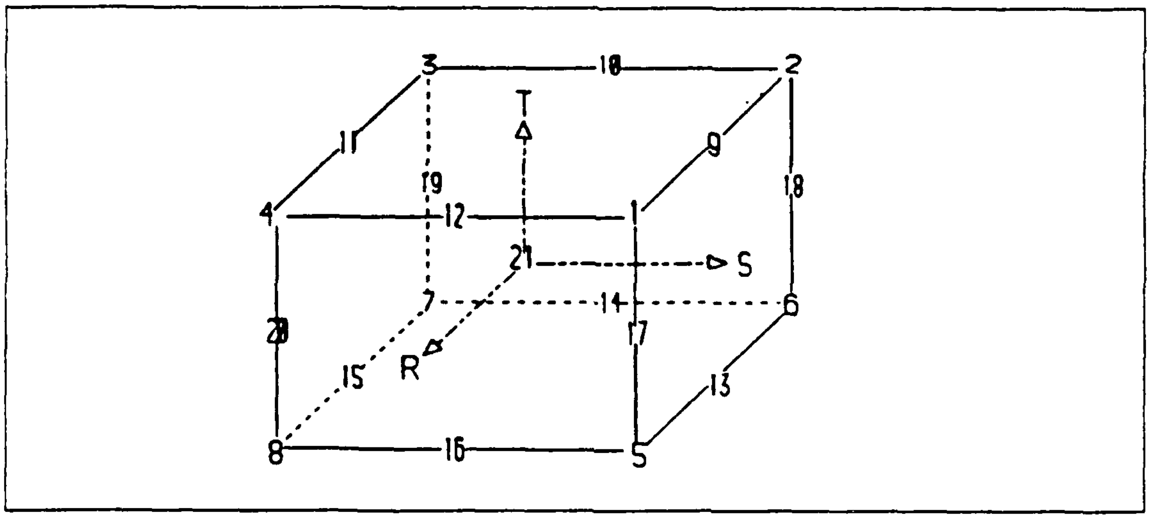


Figure 18. **R, S and T Axes for Three Dimensional Heat Transfer Elements.**

Note 5: The property values are as follows:

Flux-Flux per unit area.
Convection-Constant heat transfer coefficient (ignored if temperature dependent).
Radiation - Product of emissivity by Stephan-Boltzman constant.

Note 6: The ambient temperature is ignored for the flux boundary condition.

Note 7: If the boundary condition code is four (temperature dependent heat transfer coefficient), a table for the temperature dependence must be entered in free format and contain the following information:

ITEM	CONTENTS
1	Number of data pairs to be entered.
2	Temperature 1.
3	Heat transfer coefficient 1.
4	Temperature 2.
5	Heat transfer coefficient 2.
ETC.	ETC.
2N	Temperature N.
2N + 1	Heat transfer coefficient N.

PTEMP

This operation prints the nodal temperatures of a heat transfer system.

SYMC

This operation forms the symmetric conductance matrix for heat transfer systems.

TOL,M1

This operation sets the solution convergence tolerance to the value found in the 1 X 1 matrix named M1.

USYMC

This operation forms the unsymmetric conductance matrix for heat transfer systems.

3. Examples

There are several example heat transfer problems available on the online demonstration facility including three dimensional problems and transient problems. Only the following two problems will be discussed and provided here.

a. Hollow Cylinder With Circumferential Heating Strips

This problem was published by Roberts [Ref. 3: pp. 25-26] and is included here as an example of the use of the heat transfer commands. The problem consists of a hollow cylinder with a four inch outer diameter and a three inch inner diameter which was subjected to an axial forced convection condition in a wind tunnel by Professor P. F. Pucci of the Naval Postgraduate School. There were 16 one-quarter inch wide heating strips equally spaced over 180 degrees of the outer surface as shown below in Figure 19.

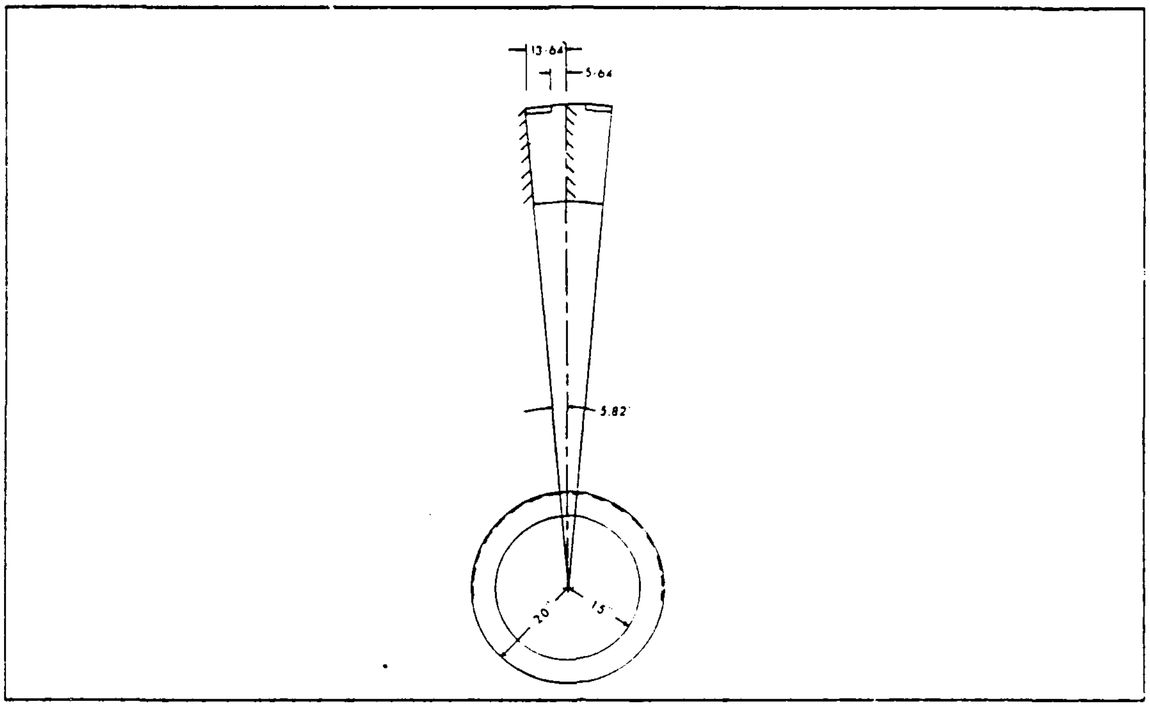


Figure 19. Hollow Cylinder.

The heating strips were maintained at 160° F. The rest of the parameters involved in this problem are as follows:

$T_{ambient}$	60° F
ρ	*LBM/FT*^{3}
C_p	0.6 BTU/(LBM °F)
$k_x = k_y$	0.1 BTU/(LBM FT °F)
h	10 BTU/(LBM *FT*^{2}°F)

The mesh consists of 99 nodes and 80 elements of four nodes each.

(1) Input Data File.

```
START
PROMPT
HTXFR
99 80 1
2 1 4
COORD
1 .1666667 5.819103 2 1
11 -.0013019 0 23
2 .1666667 4.252421 2 1
11 -.0013019 0 24
3 .1666667 3.469080 2 1
11 -.0013019 0 25
```

```
4 .1666667 2.909551 2 1
11 -.0013019 0 26
5 .1666667 2.461928 2 1
11 -.0013019 0 27
6 .1666667 2.238116 2 1
11 -.0013019 0 28
7 .1666667 2.014305 2 1
11 -.0013019 0 29
8 .1666667 1.790493 2 1
11 -.0013019 0 30
9 .1666667 1.342870 2 1
11 -.0013019 0 31
10 .1666667 0.783341 2 1
11 -.0013019 0 32
11 .1666667 0 2 1
11 -.0013019 0 33
34 .161458 5.819103 2 1
11 -.003906 0 56
35 .161458 4.252421 2 1
11 -.003906 0 57
36 .161458 3.469080 2 1
11 -.003906 0 58
37 .161458 2.909551 2 1
11 -.003906 0 59
38 .161458 2.461928 2 1
11 -.003906 0 60
39 .161458 2.238116 2 1
11 -.003906 0 61
40 .161458 2.014305 2 1
11 -.003906 0 62
41 .161458 1.790493 2 1
11 -.003906 0 63
42 .161458 1.342870 2 1
11 -.003906 0 64
43 .161458 0.783341 2 1
11 -.003906 0 65
44 .161458 0 2 1
11 -.003906 0 66
67 .147135 5.819103 2 1
11 -.0110675 0 89
68 .147135 4.252421 2 1
11 -.0110675 0 90
69 .147135 3.469080 2 1
11 -.0110675 0 91
70 .147135 2.909551 2 1
11 -.0110675 0 92
71 .147135 2.461928 2 1
11 -.0110675 0 93
72 .147135 2.238116 2 1
11 -.0110675 0 94
73 .147135 2.014305 2 1
11 -.0110675 0 95
```

```
74 .147135 1.790493 2 1
11 -.0110675 0 96
75 .147135 1.342870 2 1
11 -.0110675 0 97
76 .147135 0.783341 2 1
11 -.0110675 0 98
77 .147135 0 2 1
11 -.0110675 0 99
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
ELCON
1 12 13 2 1 1 1
1 1 1 1 1 0 10
11 23 24 13 12 1 1
1 1 1 1 1 0 20
21 34 35 24 23 1 1
1 1 1 1 1 0 30
31 45 46 35 34 1 1
1 1 1 1 1 0 40
41 56 57 46 45 1 1
1 1 1 1 1 0 50
51 67 68 57 56 1 1
1 1 1 1 1 0 60
61 78 79 68 67 1 1
1 1 1 1 1 0 70
71 89 90 79 78 1 1
1 1 1 1 1 0 80
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
CTEMP
1 6 1 160
89 99 1 60
0 0 0 0 0 0 0 0 0 0
PROP
1 2
.1 .1 .6 70 0
4 1 5 0
6 2 2 10 60
7 2 2 10 60
8 2 2 10 60
9 2 2 10 60
10 2 2 10 60
PROF
SYMC
FORM
CALC
PTEMP
PROMPT
GRAPH
YES
TITLE,3
PLHX
READ,5
```

b. Chimney Problem

This problem was solved in Kitchin's thesis using the Automatic Dynamic Incremental Nonlinear Analysis (ADINA) code [Ref. 7]. The problem was run here using CALNPS to verify the changes to CALNPS. The geometry of the problem is shown in Figure 20. The inside surface is held constant at 650°C and the outside is constant at 150°C. The results from CALNPS were in exact agreement with ADINA.

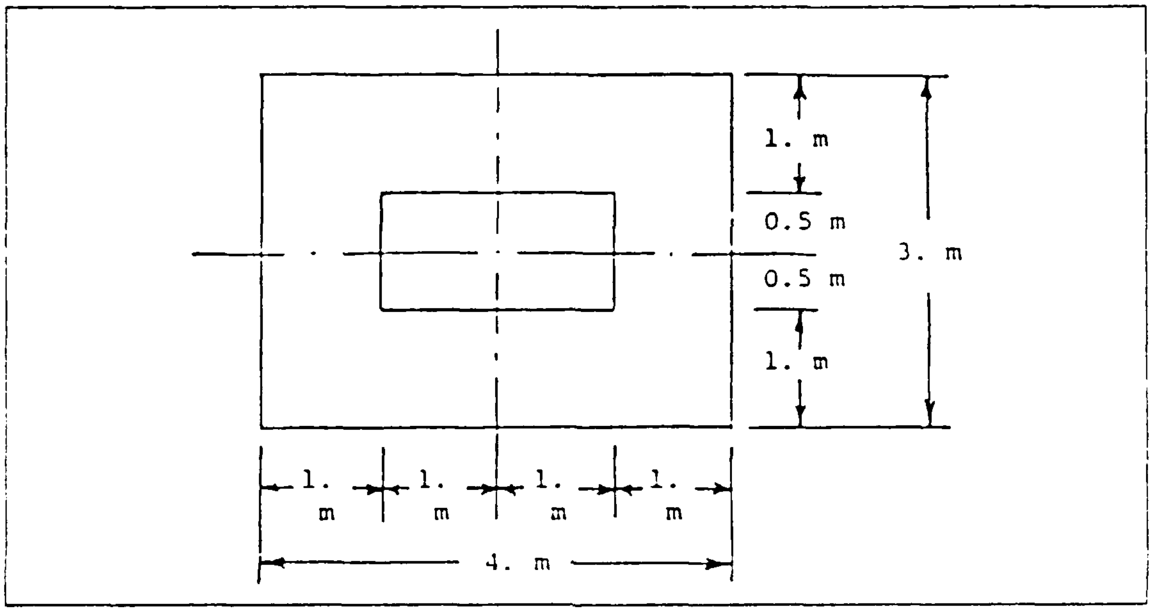


Figure 20. Geometry of Chimney Problem.

(1) Input Data File.

```
START
PROMPT
LABEL,1
                    ** CHIMNEY PROBLEM **
HTXFR
21 3 1
2 1 9
COORD
1 2 1.5 1 1
1 -.5 0 3
4 2 1.25 1 1
1 -.5 0 6
7 2 1 1 1
1 -.5 0 11
12 2 .5 1 1
1 -.5 0 16
17 2 0 1 1
1 -.5 0 21
```

```
0 0 0 0 0 0 0 0
ELCON
1 9 7 1 3 8 4 2 6 5 1 0
2 19 17 7 9 18 12 8 14 13 1 0
3 21 19 9 11 20 14 10 16 15 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0
CTEMP
1 7 3 150
12 12 0 150
17 21 1 150
3 9 3 650
10 11 1 650
0 0 0 0 0 0
PROP
1 2
4.16 4.16 .2 0 0 4 1 0 0
PROF
SYMC
FORM
CALC
WRITE,1
LABEL,3
                 * CHIMNEY PROBLEM TEMPERATURES *
                 * NINE NODES PER ELEMENT       *
                 * EXTERIOR SIDES HELD CONSTANT AT 150°C *
PTEMP
GRAPH
YES
TITLE,1
PLHX
WRITE,6
READ,5
```

(2) *Output Data File.* The output data file is shown here to demonstrate how a portion of the output can be selected. The output that occurs between the WRITE statements is sent to a file which can later be printed.

```
LABEL,3
                 * CHIMNEY PROBLEM TEMPERATURES *
                 * NINE NODES PER ELEMENT *
                 * EXTERIOR SIDES HELD CONSTANT AT 150°C *
PTEMP
NODAL TEMPERATURES
   NODE          TEMP
    1            0.15000E+03
    2            0.37514E+03
    3            0.65000E+03
    4            0.15000E+03
    5            0.36855E+03
    6            0.65000E+03
    7            0.15000E+03
```

```
     8          0.34000E+03
     9          0.65000E+03
    10          0.65000E+03
    11          0.65000E+03
    12          0.15000E+03
    13          0.25300E+03
    14          0.34267E+03
    15          0.38803E+03
    16          0.39347E+03
    17          0.15000E+03
    18          0.15000E+03
    19          0.15000E+03
    20          0.15000E+03
    21          0.15000E+03
  WRITE,6
```

F. GRAPHICS OPERATIONS

1. Description

The graphics capabilities of CALNPS were expanded considerably during the course of this work. The graphics operations must first be initialized with the command, GRAPH. When this command is issued the user must respond to a prompt for the size of the output desired. The small option provides a printout that is 6 3/4 inches by 4 5/8 inches which is ideal for use in reports. The large option is 10 inches by 7 1/2 inches. High resolution graphics is available using the LA75 printer and provides a high quality product. The TITLE command allows the user to input up to three lines of title data. The commands PLHX and PLST plot two and three dimensional heat transfer and structural analysis meshes respectively. XYPLOT allows the user to plot curves within CALNPS from a prepared data file.

2. Command Specifications

GRAPH

This operation initializes the graphics package for CALNPS. If the user is using an IBM terminal, then the following information must be provided as a response to a prompt:

Terminal Code	Terminal
1	Plot-10 Compatible.
2	IBM 3277 Dual Screen.
3	IBM (No graphics).

This response is not required on the VAX computer.

On the VAX computer, the user has the choice of two sizes of printouts. All required entries are controlled by user prompts.

PLHX,N1

This operation plots the two or three dimensional heat transfer analysis mesh. If the mesh is three dimensional, N1 specifies the plane of observation.

N1 = 1	X-Y plane.
N1 = 2	Y-Z plane.
N1 = 3	X-Z plane.

PLST,M1,M2,N1

This operation plots two and three dimensional structural analysis meshes.

M1 M1 is the node coordinate matrix created by the NODES operation.

M2 M2 is the element connectivity matrix. The row dimension is the number of elements and the column dimension is the maximum number of nodes plus one. It is created by the LOADI command and contains the following:

M2(K,1) = Number of nodes in element number K.
M2(K,2) = Node 1.
M2(K,3) = Node 2.
 ETC.   ETC.
M2(K,N+1) = Node N.
If the structure is made up of trusses only, M2 would be dimensioned number of trusses by three. If the structure contains plates (PLANE operation), the connectivity must follow the convention shown in Figure 7 on page 41.

N1 = 1	The presentation represents the X-Y plane.
N1 = 2	The presentation represents the Z-Y plane.
N1 = 3	The presentation represents the X-Z plane.

TITLE,N1

This operation allows the user to input up to three 15 character lines to label plots generated with the PLST and PLHX commands. The label will appear in the upper right corner of the display. This command must immediately precede the plot operation.

N1 N1 is the number of lines in the title (up to three). The default is one.

XYPLOT

This operation produces a plot on a X-Y scale of one or two curves. The data must be read from an existing data file. The name of the file is supplied by the user when requested by the appropriate prompt. All required inputs are requested by user prompts.

The data file must be arranged with data in two columns as shown below:

```
X(1)          Y(1)
X(2)          Y(2)
 .             .
 .             .
X(N)          Y(N)
```

The data should be in free format and delimited by a space or a comma. If two curves are to be plotted on the same graph, then the data for curve two should immediately follow the data for curve one in the data file.

Hardcopies of graphs may be obtained by simply answering YES to the prompt that occurs after the plotting operation is completed. When the screen plotting is completed, the user must type "CONTINUE" to return to the CALNPS domain. The hardcopy prompts will then appear. The hardcopy option creates a file in the user's directory which can be printed as any other file. The prompts request the user to specify which printer is to be used. Currently the choices are the LA75 and the LA210. Also the user must specify whether high or low resolution is to be used. The LA210 works with low resolution only. The LA75 can use high or low resolution. High resolution produces a very high quality output. Figure 21 illustrates an example set of curves plotted with the XYPLOT command. The user has the capability to label the curve and the axes.

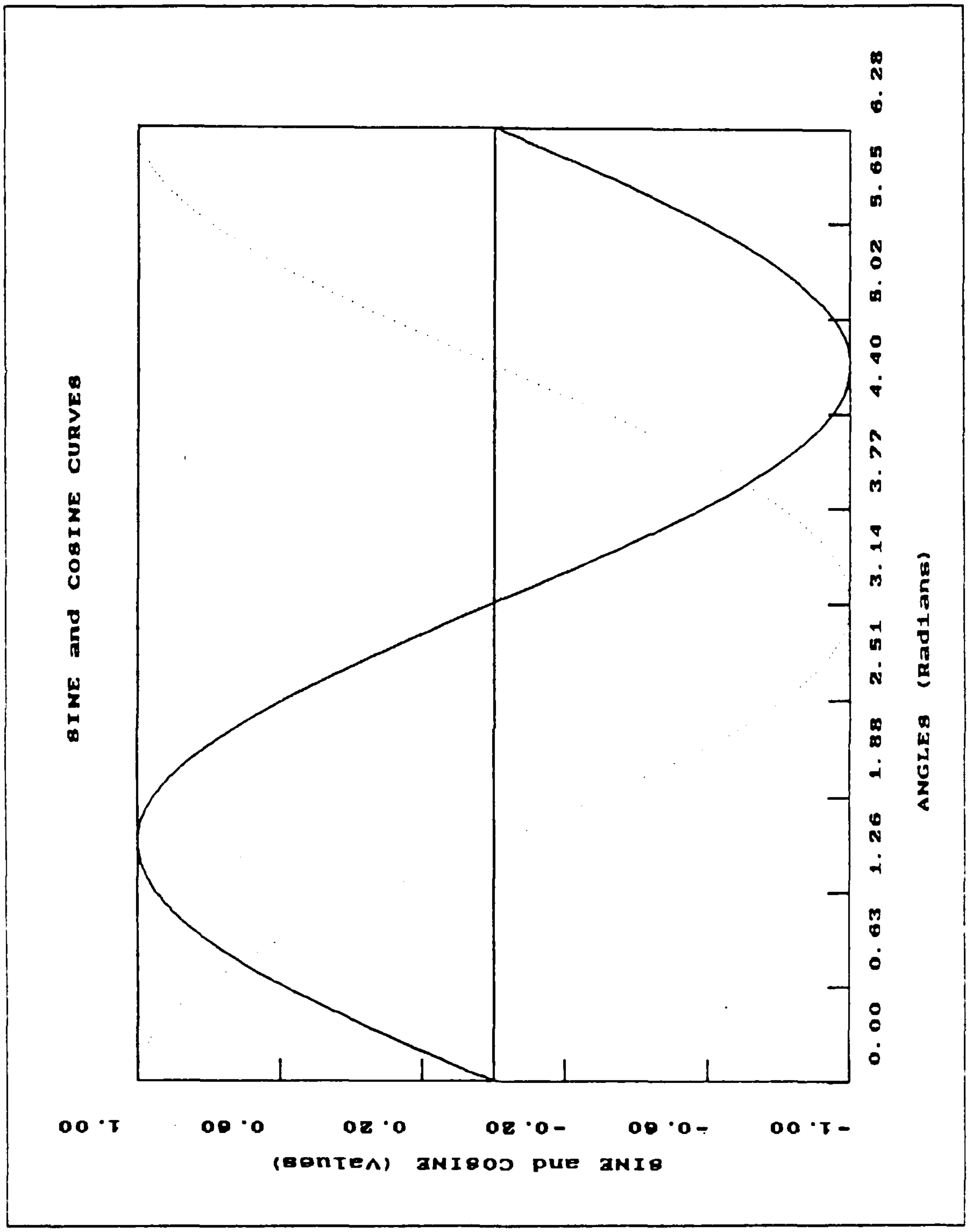


Figure 21. Sample XYPLOT Curves.

3. Examples

The examples presented here are the graphics output corresponding to example problems presented in the previous sections. The commands that produced the output are included in the input data files of the example problems. The displays in this section represent the small option of the command, GRAPH. See Figures 22 through 25.

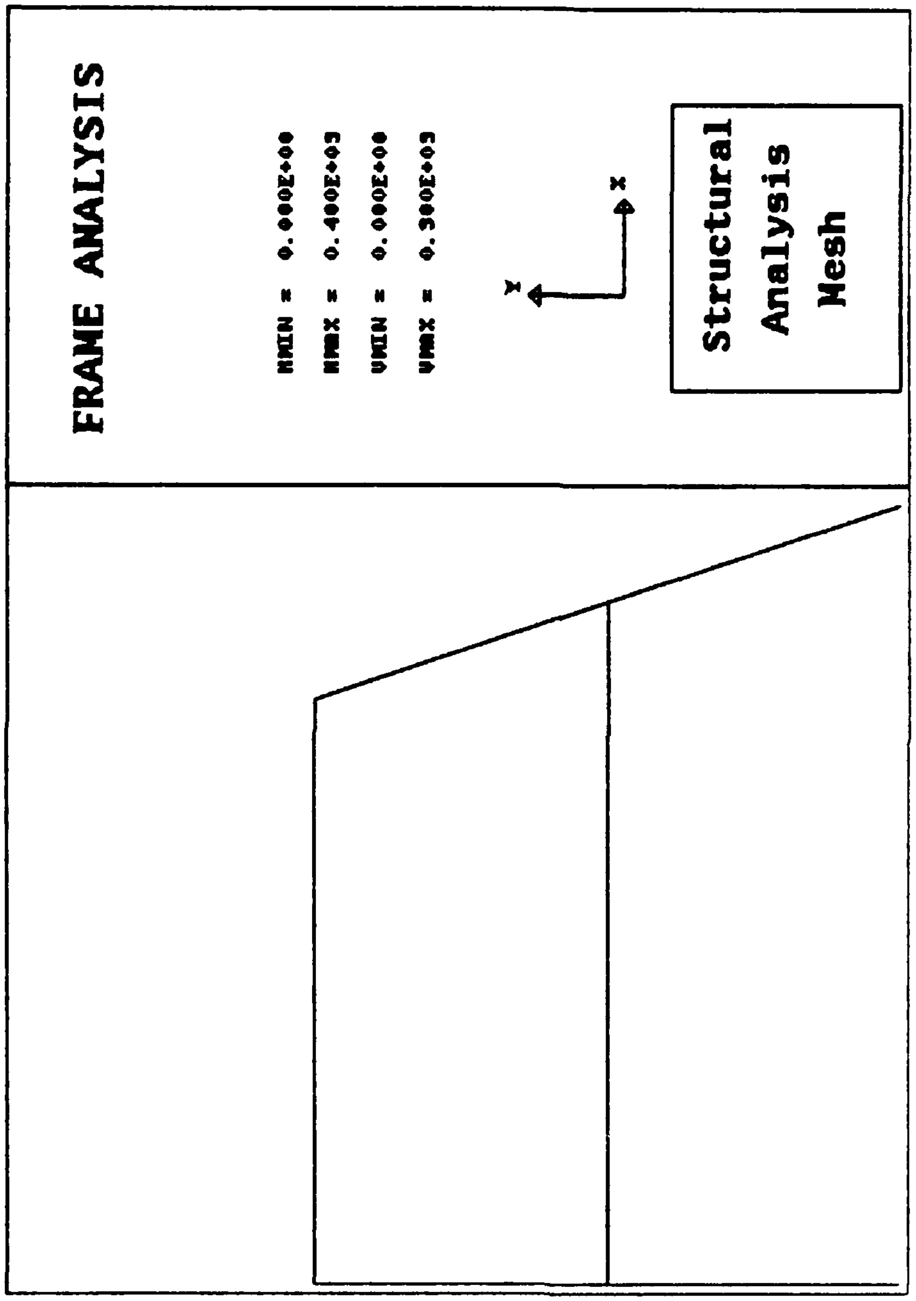


Figure 22. Frame Problem.

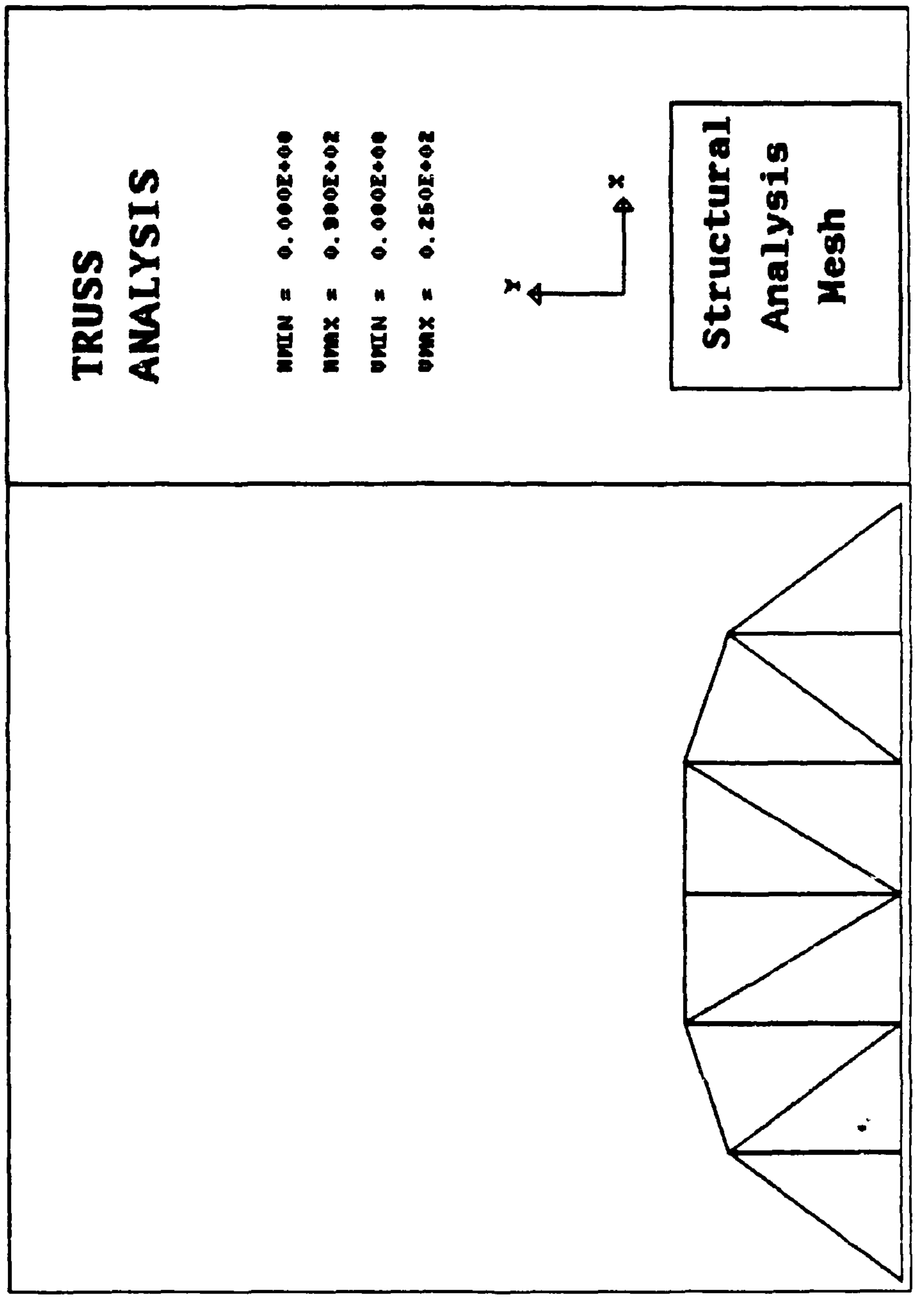


Figure 23. Truss Problem.

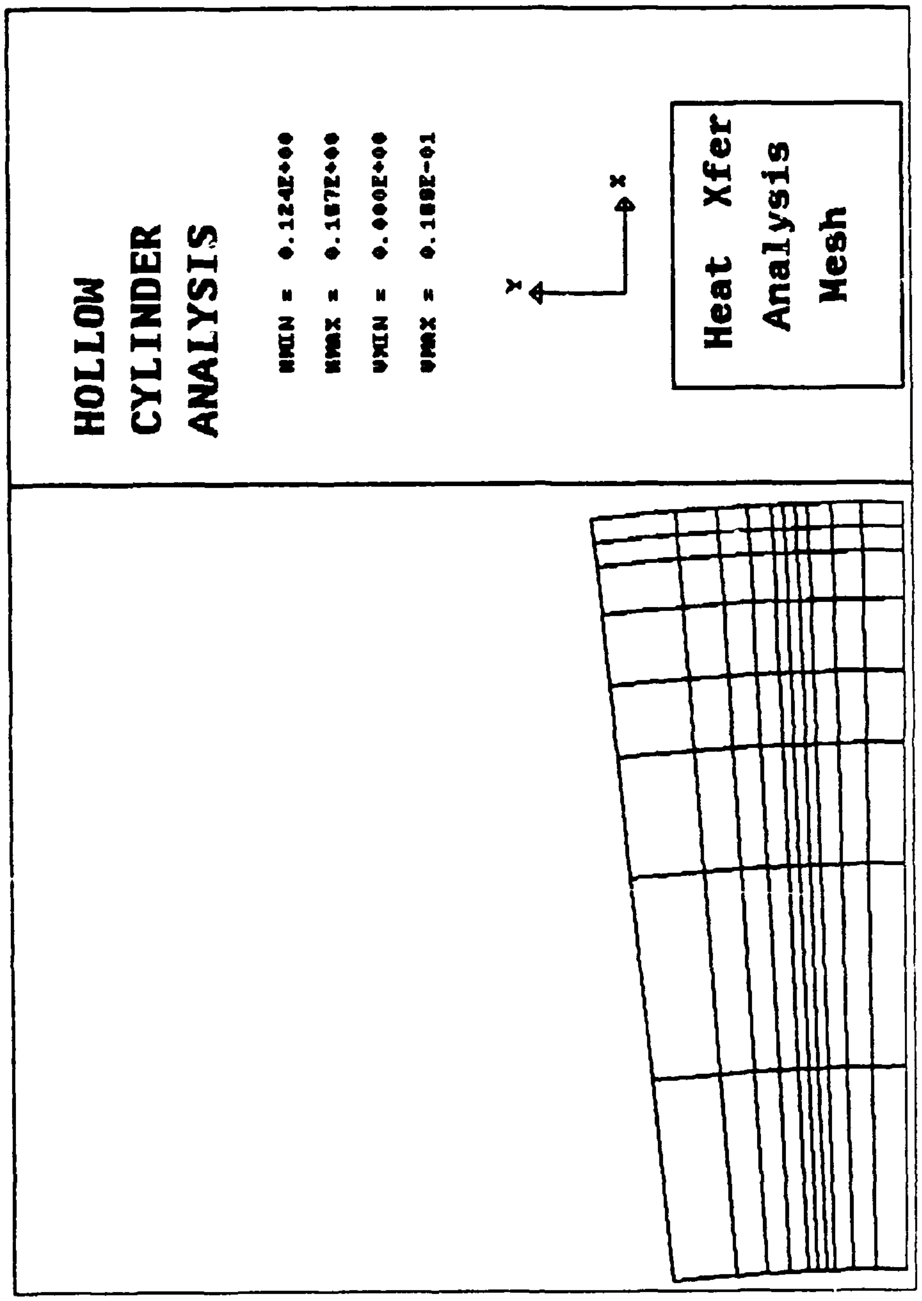


Figure 24. Hollow Cylinder Problem.

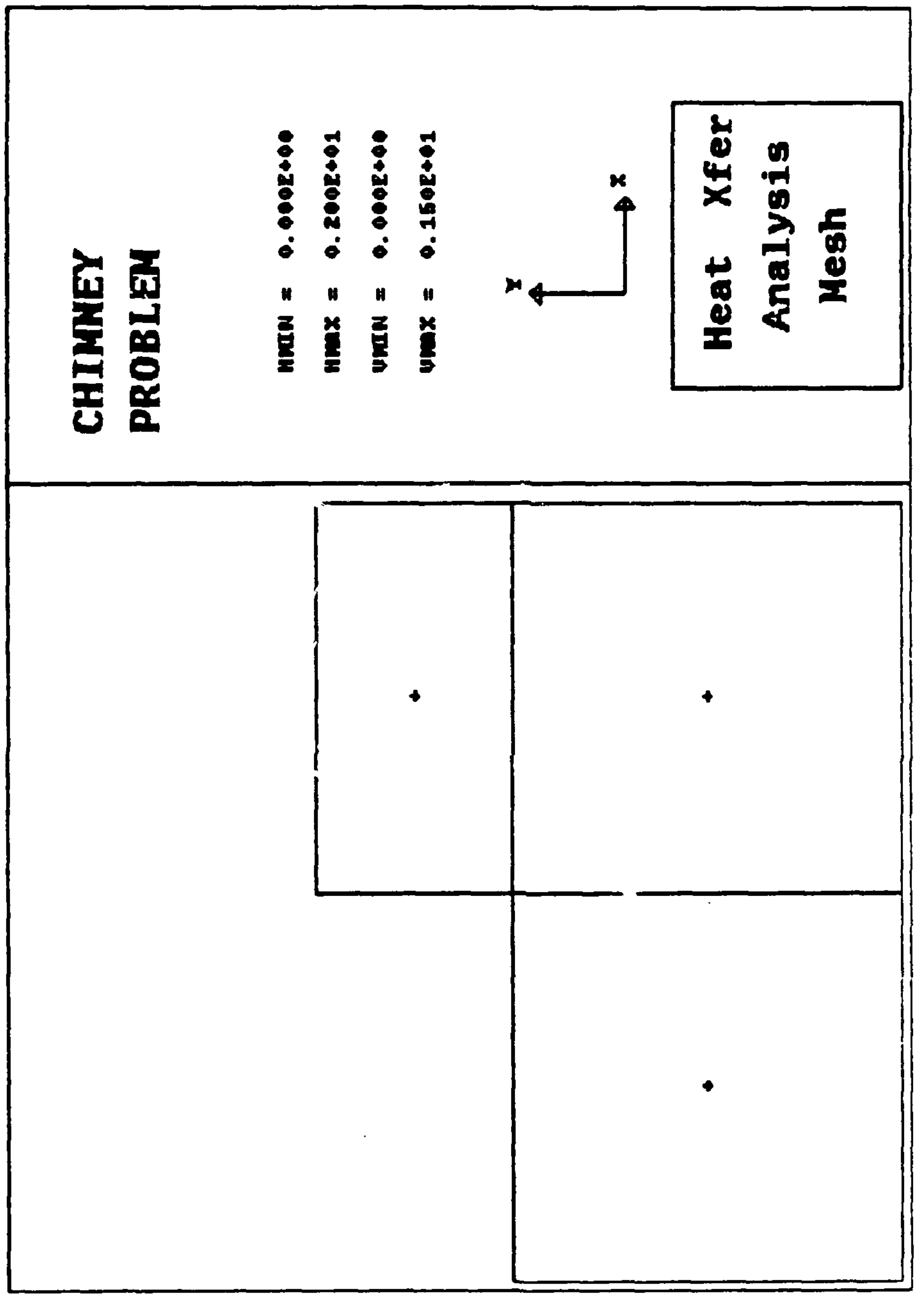


Figure 25. Chimney Problem.

G. LOOPING OPERATIONS

No changes were made to the looping operations commands during this work.

1. Description

CALNPS has a five level looping capability. The looping operation is initiated with the command, LOOP. The command, NEXT, causes CALNPS to loop back to the LOOP command. This works in the same manner as the FOR NEXT commands in BASIC programming.

CALNPS operations are normally executed sequentially and data required immediately follows the command. However, in the case of looping operations, all operations between the LOOP and NEXT commands are stored within the computer before they are executed. If operations within the loop require data, the data must be supplied in the order required after the NEXT operation. If an error is encountered during executing of a loop, the entire matrix of loop commands is deleted and the user is given the opportunity to try again. Matrices that have been modified by operations successfully completed while in the loop remain modified. after all loops are executed the computer storage required for these operations is released.

2. Command Specifications

<u>**LOOP,N1**</u>

This operation initiates the looping process. N1 is the number of times the loop is to be executed. Associated with each loop operation there must be a corresponding NEXT operation which signifies the end of a loop and causes the return to the beginning of the loop if the loop has not been executed the required number of times. The following is a possible series of looping operations:

```
LOOP,5 ..First level loop executed five times. ← ← ← ← ←
......                                                  ↑
......                                                  ↑
LOOP,2 ..Second level loop executed two times. ← ←      ↑
......     (ten times total)                      ↑     ↑
......                                            ↑     ↑
NEXT → → → → → → → → → → → → → → → → →            ↑
.....                                                   ↑
......                                                  ↑
NEXT → → → → → → → → → → → → → → → → → → → →
```

NEXT,M1

This operation signifies the end of a loop. It is apparent which LOOP and NEXT lines are associated if there are an equal number of each. The operation NEXT,M1 will cause the loop to terminate if the first term in the matrix M1 is negative.

SKIP,M1

This operation will cause the next N1 operations to be skipped if the first term of the matrix named M1 is negative for this level of looping.

H. USER SUPPLIED OPERATIONS

As mentioned in Chapter II, the USERA and USERB commands are available for the user to add additional subroutines. To execute this option the user must have the capability to recompile CALNPS and to relink CALNPS with the graphics libraries. Therefore this option is not available to all users. The procedure for using this option and the theory behind it is discussed in great detail in Elliott's thesis [Ref. 2: pp. 40-43].

V. RECOMMENDATIONS AND CONCLUSIONS

The objectives stated in chapter one were met beyond expectation. At the beginning of this work several of the CALNPS commands did not work at all. The graphics options were limited and did not work as intended on the VAX computer. The user's manual and the online "HELP" facility were obsolete. CALNPS is now completely functional on the VAX. The graphics capabilities are of a high enough quality for direct inclusion in thesis work.

Chapter four of this work serves as a revised user's manual for CALNPS. With this manual a new user can quickly become familiar enough with the commands available to solve the problem at hand. The example files included and the files available on the online demonstration facility work in conjunction with the command specifications to demonstrate how the commands of CALNPS work together to solve structural and heat transfer problems.

CAL was originally designed as an instructional tool for structural analysis. The current version of CALNPS is not only valuable as an instructional tool but is readily available as a research tool for complex problems. The following are recommendations for future use of CALNPS:

- Use CALNPS in the classroom to solve statics, dynamics and heat transfer problems.
- Make students more aware of the capabilities of CALNPS by introducing it in the computer course.
- Use CALNPS in thesis research.
- Expand CALNPS as necessary to complete thesis research. Refer to Appendix A for a discussion of how to modify CALNPS.
- Expand CALNPS to include Fluid Analysis capabilities.

CALNPS has the following advantages over other software packages even though there are several more powerful packages:

- CALNPS is very "user friendly."
- CALNPS is readily available on NPS systems.
- CALNPS can readily be modified at NPS to perform other functions as desired.

CALNPS has not been utilized to its potential due to the problems addressed in Chapter III. This thesis work has corrected these problems and the use of CALNPS should be expanded.

APPENDIX A. CALNPS MODIFICATION AND TROUBLESHOOTING GUIDE

The purpose of this appendix is to provide a guide for further modification of CALNPS. CALNPS is relatively simple to modify to the user who is familiar with the internal workings of the program, however it is very time consuming to gain that familiarity. The attempt here is to provide the information gained during this thesis work to minimize the future modification efforts.

A. CALNPS VARIABLES

The list of variables provided here is not a complete list of variable definitions, but a list of the variables used in this work. As future modifications are made additions to this list should be made.

AENGY	Energy.
D(1)	Conductivity in x-direction.
D(2)	Conductivity in y-direction.
D(3)	Specific heat.
D(4)	Mass density.
D(5)	Heat generation per unit volume.
IEL	Element type number (two or three dimensional).
IGRAF	Determines if the graphics package has been initialized.
II	Name of array being searched for.
INCNR	Number of times the increment operation is performed.
INOL	Code indicating if any material property is temperature dependent.
IOP	Number of operations (commands) in the current group.
IPR	Number of integer words required to store an element.
IPRMT	Determines whether user prompts will be printed or suppressed.
ISMALL	Determines size of graphics display.
ISYS	Coordinate system of input data.
ITERM	1 = PLOT10 compatible terminal; 2 = IBM terminal.
ITYPE	Determines whether the problem being executed is a heat transfer or a structural problem in the SAVE2 subroutine.
IVECT	Indicates presence of an increment generation vector.

KAT	Plane geometry = 1; Axisymmetry = 2.
KBCOND(I)	Boundary condition code.
KEL(I)	Element number.
KLINE(I)	Line number code.
L(NA)	Used to pass matrix NM to and from called subroutine.
LBYTE	Number of bytes per integer word.
MA	Material set number.
MODE	Selects interactive or batch mode (.NE. 1 = batch; 1 = interactive)
NA	Index in L array for first element of array.
NC	Number of columns in a matrix.
NDM	Spatial dimension.
NDP	Number of computer words used by a real variable.
NDF	Number of unknowns per node.
NEL	Number of elements in an array.
NEN	Number of nodes per element.
NERR	File device number for error messages.
NGP	Number of integration points.
NH	Number of words required to contain six Hollerith characters.
NINC	Node number increment for self generation.
NLBC	Number of lines with specified boundary conditions.
NLST	Last node to be generated.
NM	Array name as assigned by user.
NR	Number of rows in a matrix.
NREAD	Read file device number.
NROW	Array containing the standard element connectivity.
NSAVE	File device number for save file.
NT	File device number for scratch tape.
NUML	Number of elements.
NUMNDP	Number of nodes.
NUMMAT	Number of material sets.
NWRITE	Write file device number.
PROPB(I,I)	Property value.

PROPB(I,2)	Ambient temperature.
SHP(1,I)	X derivative of shape function.
SHP(2,I)	Y derivative of shape function.
SHP(3,I)	Shape functions for 2-D element or Z derivative of shape function for 3-D element.
SHP(4,I)	Shape functions for 3-D heat transfer elements.
TMIN	Minimum time for proportional load table in heat transfer problems.
TMAX	Maximum time for proportional load table in heat transfer problems.
WG	Integration weight.
X(1,I)	X nodal coordinates.
X(2,I)	Y nodal coordinates.
X(3,I)	Z nodal coordinates.
XINC	X increment for self generation of nodes.
XS	Jacobian array.
XSJ	Jacobian determinant.
YINC	Y increment for self generation of nodes.
ZINC	Z increment for self generation of nodes.

B. PROCEDURE FOR THE ADDITION OF A NEW COMMAND

This section provides the procedure that should be followed to create a new command in the CALNPS code. This procedure is consistent with the original design. The procedure as stated here is relatively simple, however, it is not a simple task for someone unfamiliar with the inner workings of CALNPS to establish.

The programmer must first decide which group of subroutines that the new command belongs to. Within the appropriate group, the "DIMENSION IOP" statement should be located. This dimension establishes the number of commands currently in that particular group. This dimension should be increased to accommodate the new commands. A list of data statements follows the dimension statement. These data statements define the commands to CALNPS. A sample statement is as follows:

```
DATA IOP(1,42),IOP(2,42),IOP(3,42) 4HCUBI,4HC   ,4H
```

This statement defines the command, CUBIC, as operation number 42. The number of operations in the group must then be increased as appropriate. This is done by the NUMOP=42 statement (for this case). CALNPS branches to the appropriate subroutines for the command given by means of the "computed goto" statement. This must be adjusted for the new command(s). Finally the subroutines to accomplish the new tasks must be added.

C. SUBROUTINE FLOWPATHS FOR SELECTED COMMANDS

The subroutine flowpaths of some of the more complex commands to trace through CALNPS are presented here. The subroutines operate with numerous flags that distinguish between the different conditions that apply. The most important flags are identified in the flowpaths presented in Figures 26 through 34 on the following pages. The numbers on the charts indicate the sequence of events that apply to the corresponding subroutine.

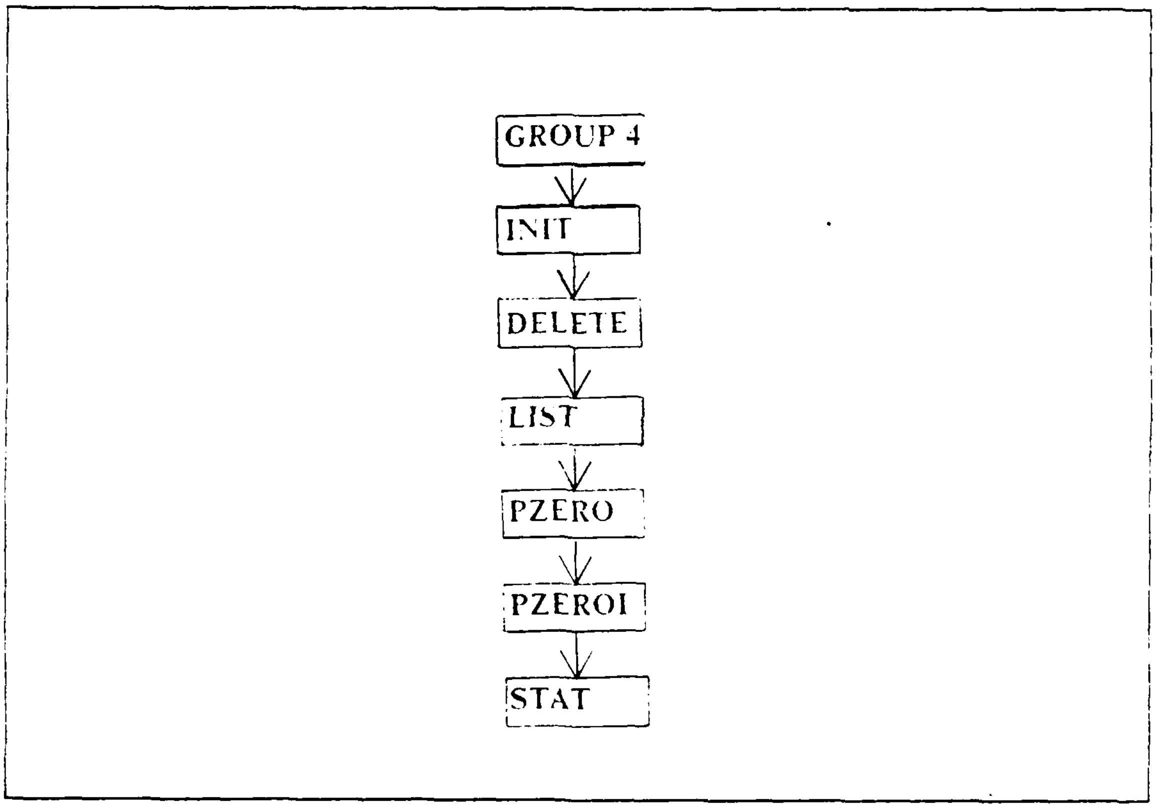


Figure 26. Flowpath for HTXFR Command.

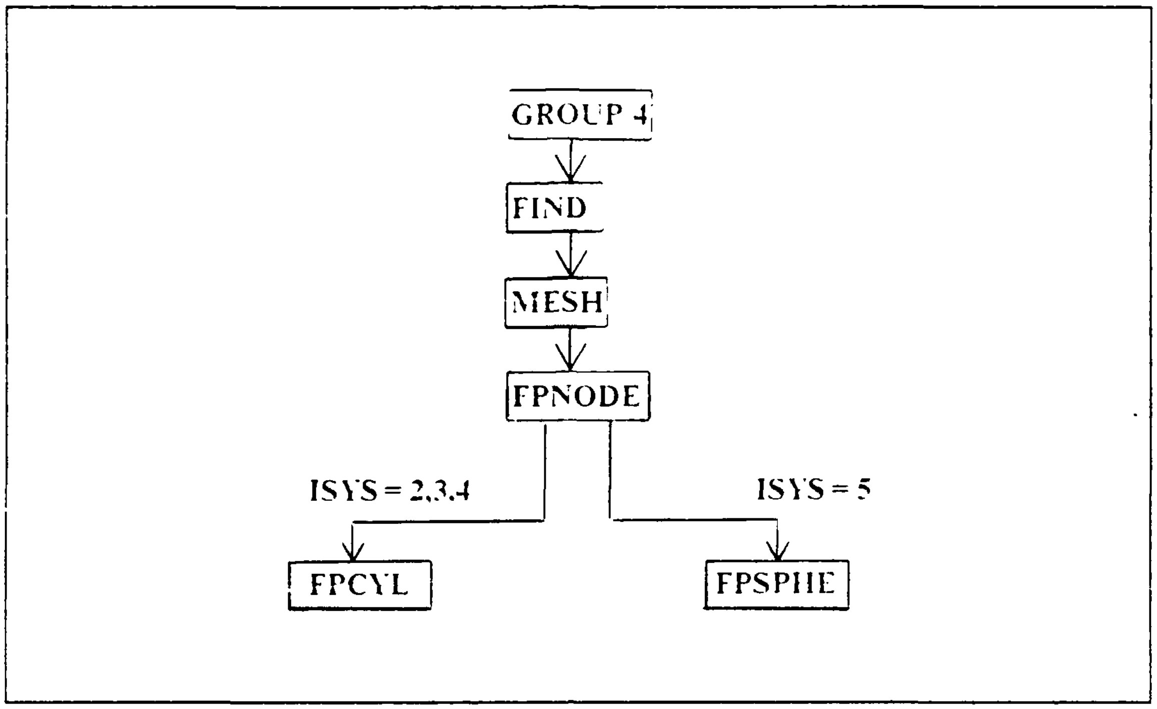


Figure 27.   Flowpath for COORD Command

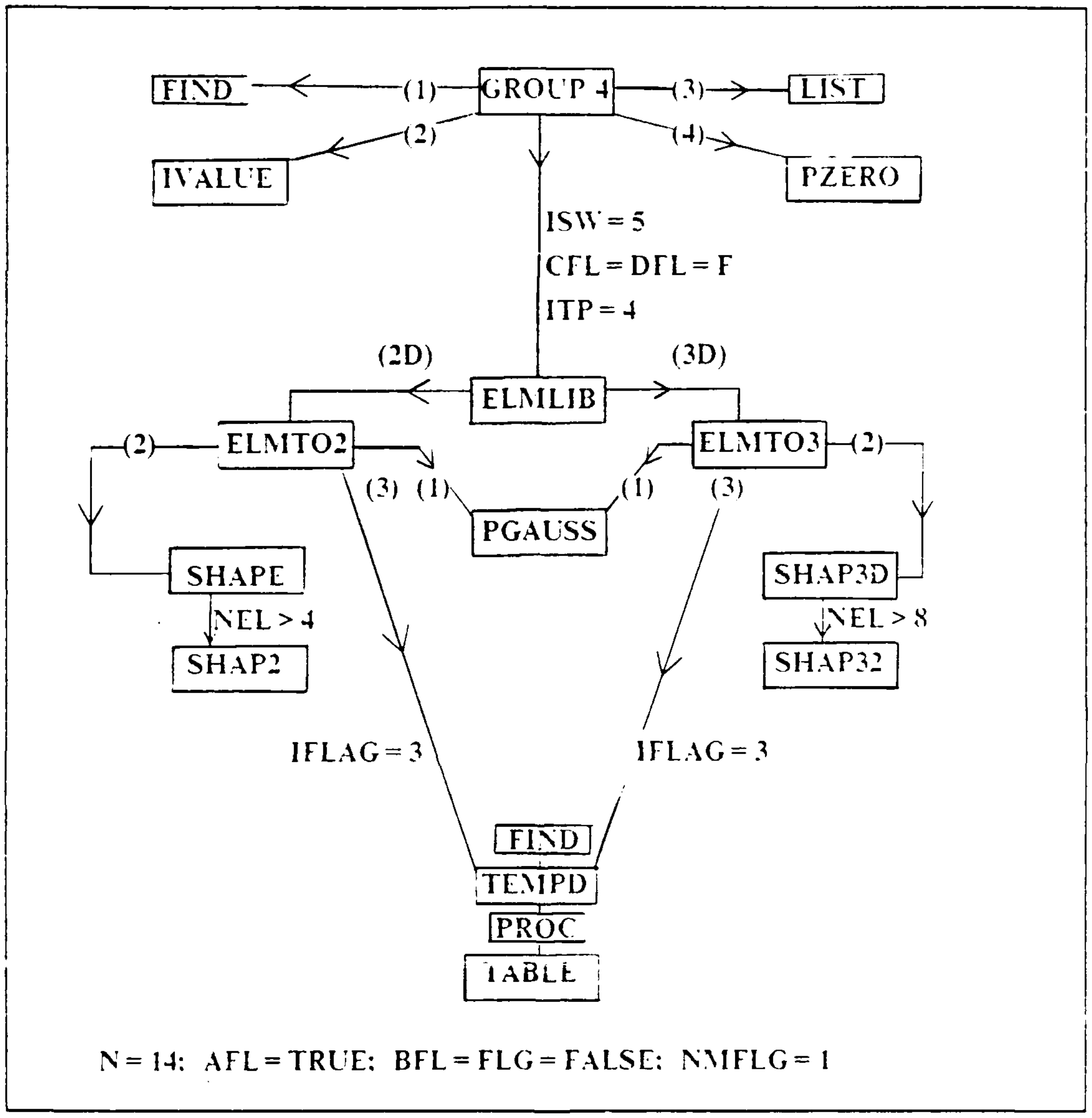


Figure 28. Flowpath for CCAP Command.

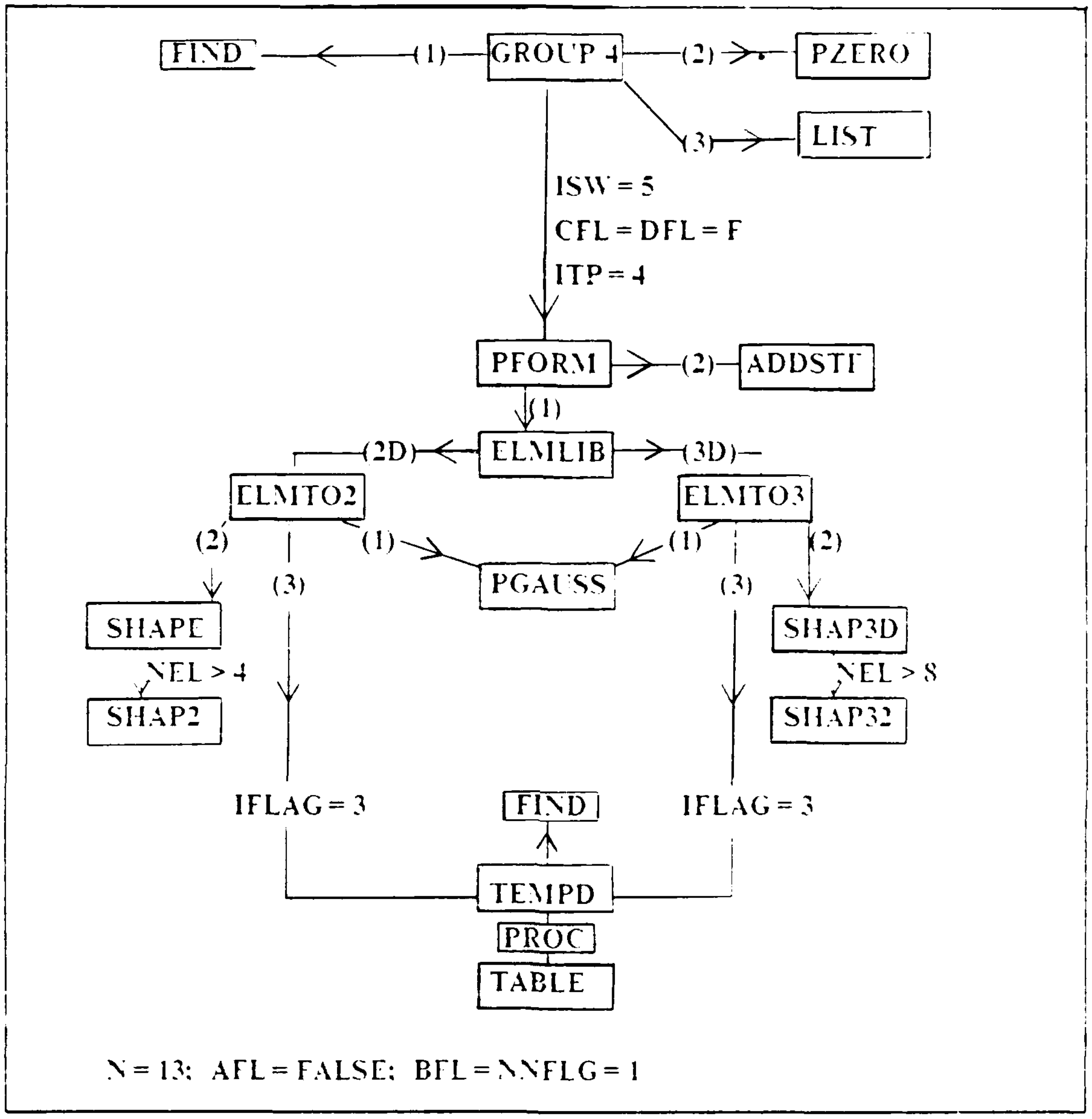


Figure 29. Flowpath for LCAP Command.

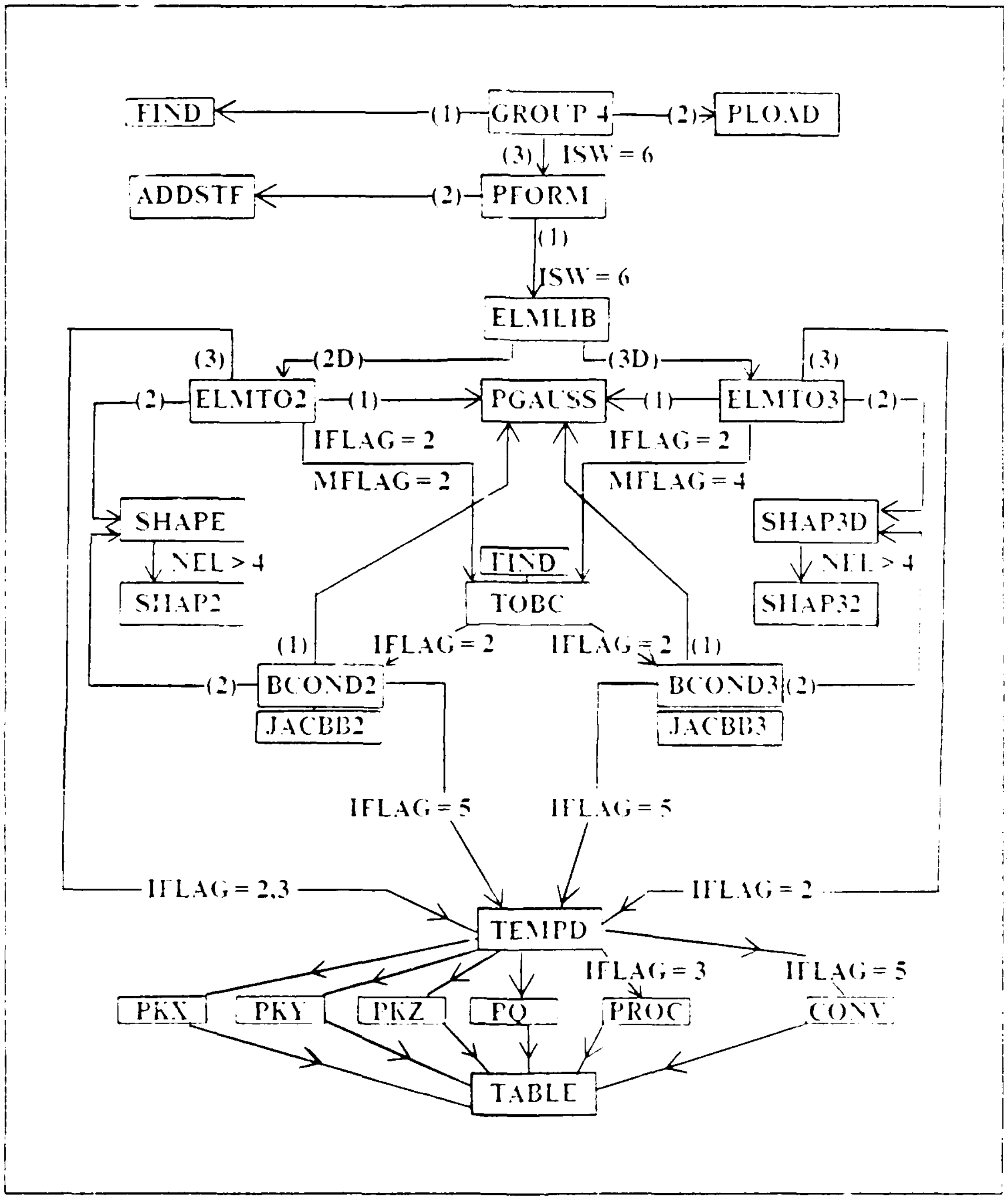


Figure 30. Flowpath for FORM Command.

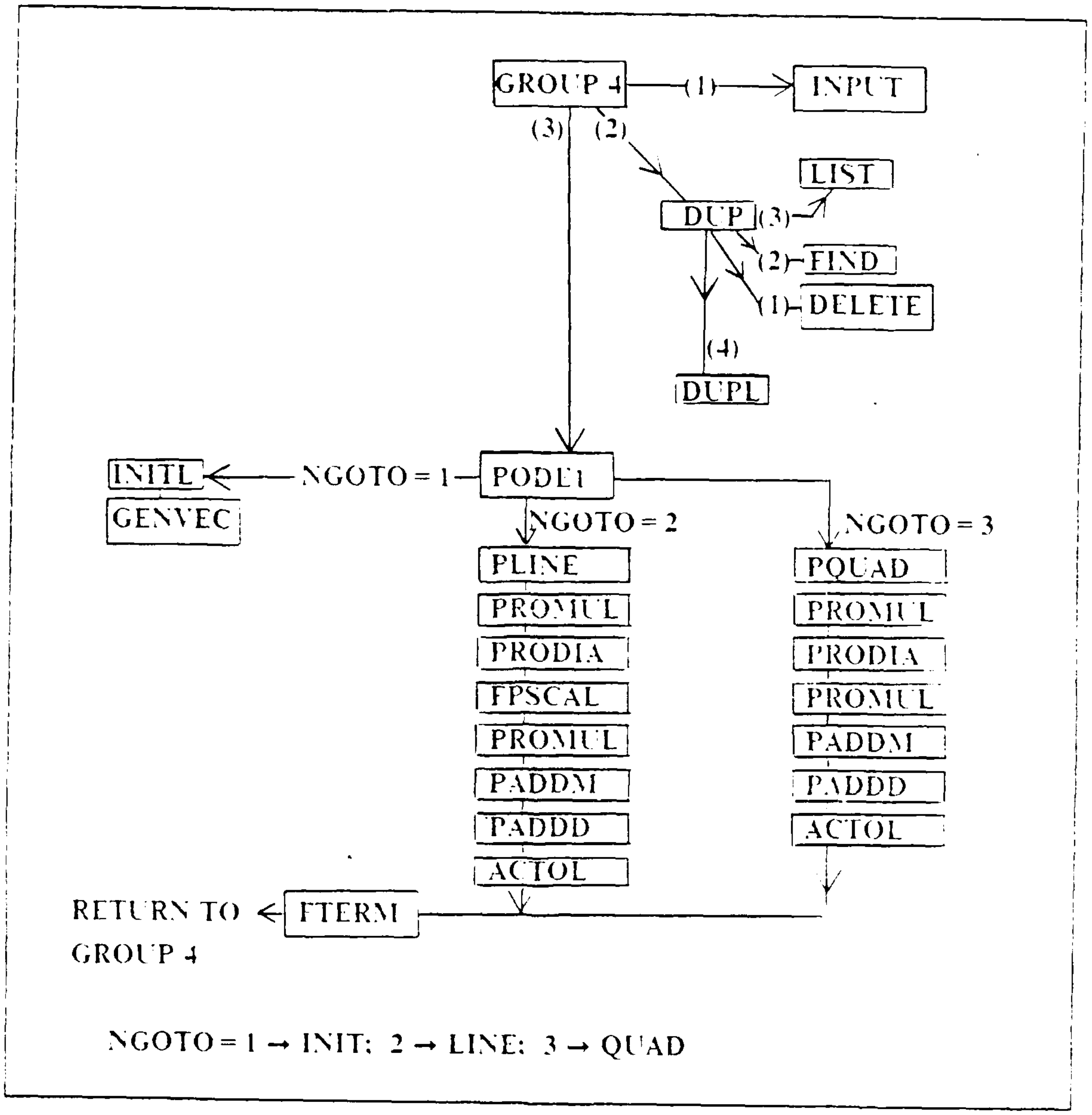
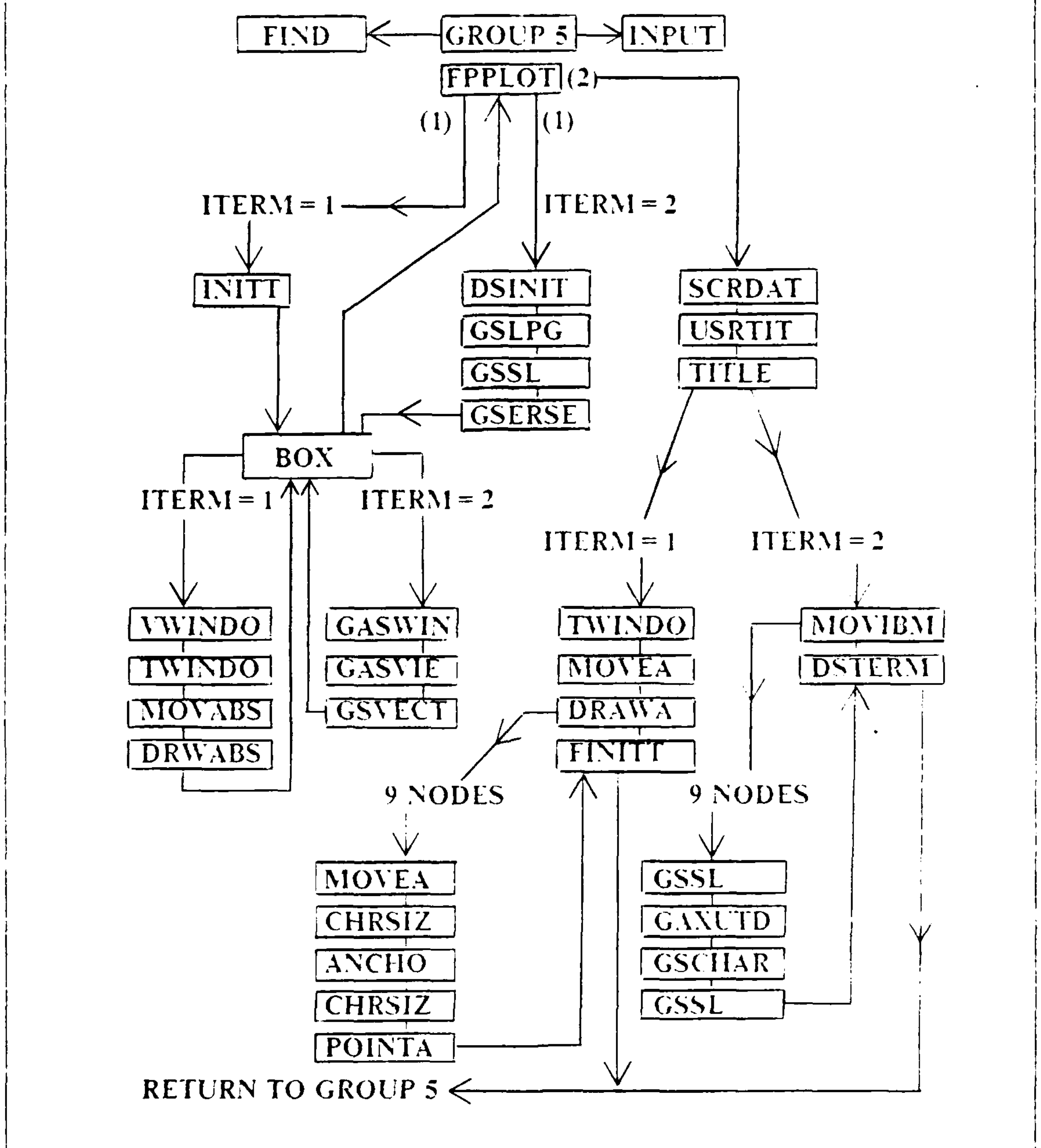


Figure 31. Flowpath for ODE Command.



Substitute FP3PLOT for FPPLOT for 3-D.
Substitute CLPLOT for FPPLOT for structural problems.

Figure 32. Flowpath for PLHX and PLST Commands.

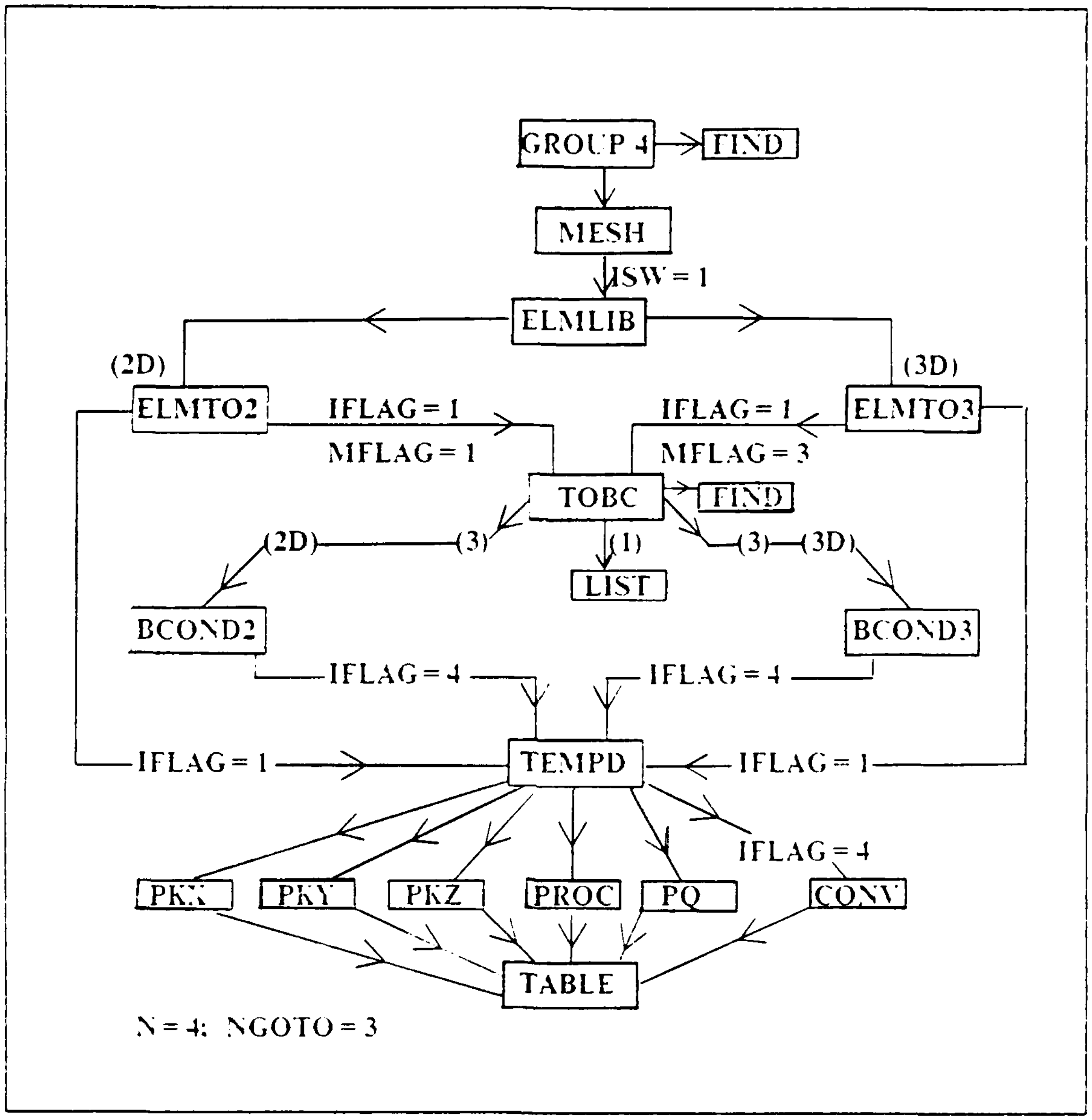


Figure 33. Flowpath for PROP Command.

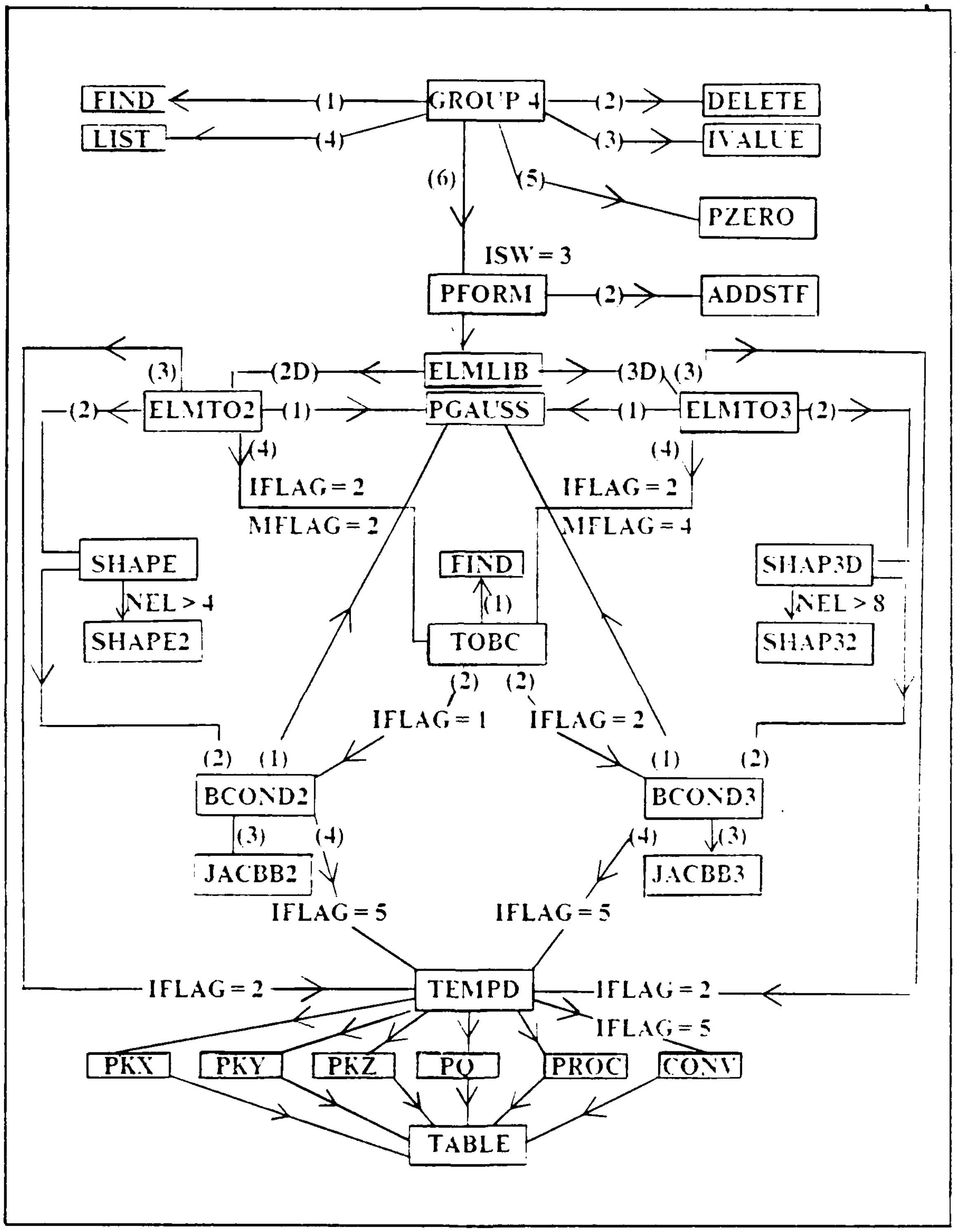


Figure 34. Flowpath for SYMC and USYMC Commands.

LIST OF REFERENCES

1. Wilson, Edward L., University of California Report No. UC SESM 77-2, *CAL Computer Analysis Language for the Static and Dynamic Analysis of Structural Systems*, January, 1977.

2. Elliott, Lawrence B., *Non-numerical Applications of Computer Programming in the Construction of Problem Oriented Languages*, Master's Thesis and Engineer's Degree, Naval Postgraduate School,Monterey ,Ca., December 1979.

3. Roberts, Warren L., *Integration of Finite Element Analysis Program for Conduction Heat Transfer With Computer Analysis Language*,Master's Thesis, Naval Postgraduate School, Monterey, Ca., June 1982.

4. Danford Corporation, *Danford PLOT10 Terminal Control System Subroutine Library User Manual* ,1984.

5. Wilson, Edward L., *CAL-86 Computer Assisted Learning of Structural Analysis and the CAL SAP Development System*,University of California, Berkeley, Ca., 1986.

6. Higdon, A.,and others, *Engineering Mechanics Volume I:Statics*, 2d ed., pp.150-153, Prentice-Hall, INC., 1976.

7. Kitchin, Doyle R., *2-Dimensional Axisymmetric and 3-Dimensional Finite Element Stress Analysis of the LHA-1 Class Superheater Header*, Master's Thesis, Naval Postgraduate School, Monterey, Ca., March 1988.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Professor Gilles Cantin Code 69Ci Department of Mechanical Engineering Naval Postgraduate School Monterey, CA 93940	5
4.	Naval Engineering Curricular Office, Code 34 Naval Postgraduate School Monterey, CA 93940	1
5.	Department Chairman, Code 69 Department of Mechanical Engineering Monterey, CA 93940	1
6.	Professor Paul F. Pucci, Code 69Pc Department of Mechanical Engineering Naval Postgraduate School Monterey, CA 93940	1
9.	Professor David Salinas, Code 69Zc Department of Mechanical Engineering Naval Postgraduate School Monterey, CA 93940	1
10.	Professor Edward L. Wilson Structural Engineering Division Civil Engineering Department University of California (Berkeley) Berkeley, CA 94720	1
11.	Professor J. L. Batoz Department de Genie Mecanique Universite de Technologie 60206 Compiegne, France	1

12. Professor Guri Dhatt 1
Centre Technique de l'Informatique
Universite Laval
Quebec, Prov. de Quebec
Canada G1K 7P4

13. Professor Gilbert Touzot 1
Centre d'Informatique
Universite de Technologie
60206 Compiegne, France

14. Professor Y. S. Shin, Code 69Sg 1
Department of Mechanical Engineering
Naval Postgraduate School
Monterey, CA 93940

15. Professor Phillip Shin, Code 69 1
Department of Mechanical Engineering
Naval Postgraduate School
Monterey, CA 93940

16. Lawrence B. Elliot 1
2721 Cedar Avenue
Long Beach, CA 90806

17. LT Leonard L. Langford, Jr. 1
324 Brittan Trail
Jonesboro, GA 30236

18. Pierre Berger 1
21 Rue Des Binelles
92310 Sevres, France